DonorsChoose

DonorsChoose.org receives hundreds of thousands of project proposals each year for classroom projects in need of funding. Right now, a large number of volunteers is needed to manually screen each submission before it's approved to be posted on the DonorsChoose.org website.

Next year, DonorsChoose.org expects to receive close to 500,000 project proposals. As a result, there are three main problems they need to solve:

- How to scale current manual processes and resources to screen 500,000 projects so that they can be posted as quickly and as efficiently as possible
- How to increase the consistency of project vetting across different volunteers to improve the experience for teachers
- How to focus volunteer time on the applications that need the most assistance

The goal of the competition is to predict whether or not a DonorsChoose.org project proposal submitted by a teacher will be approved, using the text of project descriptions as well as additional metadata about the project, teacher, and school. DonorsChoose.org can then use this information to identify projects most likely to need further review before approval.

About the DonorsChoose Data Set

The `train.csv` data set provided by DonorsChoose contains the following features:

Feature	Description
`project_id`	A unique identifier for the proposed project. **Example:** `p036502`
`project_title`	Title of the project. **Examples:** • `Art Will Make You Happy!` • `First Grade Fun`
`project_grade_category`	Grade level of students for which the project is targeted. One of the following enumerated values: • `Grades PreK-2` • `Grades 3-5` • `Grades 6-8` • `Grades 9-12`
`project_subject_categories`	One or more (comma-separated) subject categories for the project from the following enumerated list of values: • `Applied Learning` • `Care & Hunger` • `Health & Sports` • `History & Civics` • `Literacy & Language` • `Math & Science` • `Music & The Arts` • `Special Needs` • `Warmth` **Examples:** • `Music & The Arts` • `Literacy & Language, Math & Science`
`school_state`	State where school is located (Two-letter U.S. postal code). **Example:** `WY`
`project_subject_subcategories`	One or more (comma-separated) subject subcategories for the project. **Examples:** • `Literacy`

Feature	Description
	• `Literature & Writing, Social Sciences`
`project_resource_summary`	An explanation of the resources needed for the project. **Example:** • `My students need hands on literacy materials to manage sensory needs!`
`project_essay_1`	First application essay*
`project_essay_2`	Second application essay*
`project_essay_3`	Third application essay*
`project_essay_4`	Fourth application essay*
`project_submitted_datetime`	Datetime when project application was submitted. **Example:** `2016-04-28 12:43:56.245`
`teacher_id`	A unique identifier for the teacher of the proposed project. **Example:** `bdf8baa8fedef6bfeec7ae4ff1c15c56`
`teacher_prefix`	Teacher's title. One of the following enumerated values: • `nan` • `Dr.` • `Mr.` • `Mrs.` • `Ms.` • `Teacher.`
`teacher_number_of_previously_posted_projects`	Number of project applications previously submitted by the same teacher. **Example:** `2`

* See the section **Notes on the Essay Data** for more details about these features.

Additionally, the `resources.csv` data set provides more data about the resources required for each project. Each line in this file represents a resource required by a project:

Feature	Description
`id`	A `project_id` value from the `train.csv` file. **Example:** `p036502`
`description`	Desciption of the resource. **Example:** `Tenor Saxophone Reeds, Box of 25`
`quantity`	Quantity of the resource required. **Example:** `3`
`price`	Price of the resource required. **Example:** `9.95`

Note: Many projects require multiple resources. The `id` value corresponds to a `project_id` in train.csv, so you use it as a key to retrieve all resources needed for a project:

The data set contains the following label (the value you will attempt to predict):

Label	Description
`project_is_approved`	A binary flag indicating whether DonorsChoose approved the project. A value of `0` indicates the project was not approved, and a value of `1` indicates the project was approved.

Notes on the Essay Data

Prior to May 17, 2016, the prompts for the essays were as follows:

- __project_essay_1:__ "Introduce us to your classroom"
- __project_essay_2:__ "Tell us more about your students"
- __project_essay_3:__ "Describe how your students will use the materials you're requesting"
- __project_essay_3:__ "Close by sharing why your project will make a difference"

Starting on May 17, 2016, the number of essays was reduced from 4 to 2, and the prompts for the first 2 essays were changed to the following:

- __project_essay_1:__ "Describe your students: What makes your students special? Specific details about their background, your neighborhood, and your school are all helpful."

your neighborhood, and your school are all helpful.

- __project_essay_2:__ "About your project: How will these materials make a difference in your students' learning and improve their school lives?"

  For all projects with project_submitted_datetime of 2016-05-17 and later, the values of project_essay_3 and project_essay_4 will be NaN.

In [4]:

```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

import sqlite3
import pandas as pd
import numpy as np
import nltk
import string
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.metrics import roc_curve, auc
from nltk.stem.porter import PorterStemmer

import re
# Tutorial about Python regular expressions: https://pymotw.com/2/re/
import string
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem.wordnet import WordNetLemmatizer

from gensim.models import Word2Vec
from gensim.models import KeyedVectors
import pickle

from tqdm import tqdm
import os

from plotly import plotly
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
from collections import Counter
```

```
C:\Users\samar\Anaconda3\lib\site-packages\gensim\utils.py:1197: UserWarning: detected Windows; al
iasing chunkize to chunkize_serial
  warnings.warn("detected Windows; aliasing chunkize to chunkize_serial")
```

1.1 Reading Data

In [2]:

```
project_data = pd.read_csv('train_data.csv')
resource_data = pd.read_csv('resources.csv')
```

In [3]:

```
print("Number of data points in train data", project_data.shape)
print('-'*50)
print("The attributes of data :", project_data.columns.values)
```

```
Number of data points in train data (109248, 17)
--------------------------------------------------
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' 'school_state'
 'project_submitted_datetime' 'project_grade_category'
 'project_subject_categories' 'project_subject_subcategories'
```

```
 'project_title' 'project_essay_1' 'project_essay_2' 'project_essay_3'
 'project_essay_4' 'project_resource_summary'
 'teacher_number_of_previously_posted_projects' 'project_is_approved']
```

In [4]:

```
print("Number of data points in train data", resource_data.shape)
print(resource_data.columns.values)
resource_data.head(2)
```

```
Number of data points in train data (1541272, 4)
['id' 'description' 'quantity' 'price']
```

Out[4]:

	id	description	quantity	price
0	p233245	LC652 - Lakeshore Double-Space Mobile Drying Rack	1	149.00
1	p069063	Bouncy Bands for Desks (Blue support pipes)	3	14.95

1.2 preprocessing of `project_subject_categories`

In [5]:

```
catogories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python:
https://stackoverflow.com/a/47301924/4084039

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-python
cat_list = []
for i in catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmth", "Care & H
unger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Scienc
e"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to replace it with ''(i
.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math &
Science"=>"Math&Science"
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing spaces
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())

project_data['clean_categories'] = cat_list
project_data.drop(['project_subject_categories'], axis=1, inplace=True)

from collections import Counter
my_counter = Counter()
for word in project_data['clean_categories'].values:
    my_counter.update(word.split())

cat_dict = dict(my_counter)
sorted_cat_dict = dict(sorted(cat_dict.items(), key=lambda kv: kv[1]))
```

1.3 preprocessing of `project_subject_subcategories`

In [6]:

```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python:
https://stackoverflow.com/a/47301924/4084039

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-string
```

```
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-python

sub_cat_list = []
for i in sub_catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmth", "Care & H
unger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Scienc
e"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to replace it with ''(i
.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math &
Science"=>"Math&Science"
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the trailing spaces
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())

project_data['clean_subcategories'] = sub_cat_list
project_data.drop(['project_subject_subcategories'], axis=1, inplace=True)

# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
my_counter = Counter()
for word in project_data['clean_subcategories'].values:
    my_counter.update(word.split())

sub_cat_dict = dict(my_counter)
sorted_sub_cat_dict = dict(sorted(sub_cat_dict.items(), key=lambda kv: kv[1]))
```

1.4 preprocessing of project_grade_category

In [7]:

```
prj_grade_cat = list(project_data['project_grade_category'].values)
# remove special characters from list of strings python:
https://stackoverflow.com/a/47301924/4084039

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-python

prj_grade_cat_list = []
for i in prj_grade_cat:
    for j in i.split(' '): # it will split by space
        j=j.replace('Grades','') # if we have the words "Grades" we are going to replace it with ''
(i.e removing 'Grades')
    prj_grade_cat_list.append(j.strip())

project_data['clean_grade'] = prj_grade_cat_list
project_data.drop(['project_grade_category'], axis=1, inplace=True)

# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
my_counter = Counter()
for word in project_data['clean_grade'].values:
    my_counter.update(word.split())

prj_grade_cat_dict = dict(my_counter)
sorted_prj_grade_cat_dict = dict(sorted(prj_grade_cat_dict.items(), key=lambda kv: kv[1]))

project_data['clean_grade'].values
```

Out[7]:

```
array(['PreK-2', '6-8', '6-8', ..., 'PreK-2', '3-5', '6-8'], dtype=object)
```

1.5 preprocessing of teacher_prefix

In [8]:

```
#tea_pfx_cat = list(project_data['teacher_prefix'].values)
tea_pfx_cat = list(project_data['teacher_prefix'].astype(str).values)
# remove special characters from list of strings python:
```

```
https://stackoverflow.com/a/47301924/4084039

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-python

##https://stackoverflow.com/questions/52736900/how-to-solve-the-attribute-error-float-object-has-no-attribute-split-in-pyth
#vectorizer.fit(project_data['teacher_prefix'].astype(str).values)

tea_pfx_cat_list = []
for i in tea_pfx_cat:
    #for j in i.split(' '): # it will split by space
    #j=j.replace('.','') # if we have the words "Grades" we are going to replace it with ''(i.e removing 'Grades')
    i=i.replace('.','') # if we have the words "Grades" we are going to replace it with ''(i.e removing 'Grades')
    i=i.replace('nan','') # if we have the words "Grades" we are going to replace it with ''(i.e removing 'Grades')
    tea_pfx_cat_list.append(i.strip())

project_data['clean_tea_pfx'] = tea_pfx_cat_list
project_data.drop(['teacher_prefix'], axis=1, inplace=True)

# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
my_counter = Counter()
for word in project_data['clean_tea_pfx'].values:
    my_counter.update(word.split())

tea_pfx_cat_dict = dict(my_counter)
sorted_tea_pfx_cat_dict = dict(sorted(tea_pfx_cat_dict.items(), key=lambda kv: kv[1]))

project_data['clean_tea_pfx'].values
```

Out[8]:

```
array(['Mrs', 'Mr', 'Ms', ..., 'Mrs', 'Mrs', 'Ms'], dtype=object)
```

1.6 Text preprocessing

In [9]:

```
# merge two column text dataframe:
project_data["essay"] = project_data["project_essay_1"].map(str) +\
                        project_data["project_essay_2"].map(str) + \
                        project_data["project_essay_3"].map(str) + \
                        project_data["project_essay_4"].map(str)
```

In [10]:

```
project_data.head(2)
```

Out[10]:

	Unnamed: 0	id	teacher_id	school_state	project_submitted_datetime	project_title	projec
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	IN	2016-12-05 13:43:57	Educational Support for English Learners at Home	My stu Englis that ar
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	FL	2016-10-25 09:22:10	Wanted: Projector for Hungry Learners	Our st arrive school lea

	Unnamed: 0	id	teacher_id	school_state	project_submitted_datetime	project_title	projec

In [11]:

```
#### 1.4.2.3 Using Pretrained Models: TFIDF weighted W2V
```

In [12]:

```
# printing some random reviews
print(project_data['essay'].values[0])
print("="*50)
print(project_data['essay'].values[150])
print("="*50)
print(project_data['essay'].values[1000])
print("="*50)
print(project_data['essay'].values[20000])
print("="*50)
print(project_data['essay'].values[99999])
print("="*50)
```

```
My students are English learners that are working on English as their second or third languages. W
e are a melting pot of refugees, immigrants, and native-born Americans bringing the gift of langua
ge to our school. \r\n\r\n We have over 24 languages represented in our English Learner program wi
th students at every level of mastery.  We also have over 40 countries represented with the
families within our school.  Each student brings a wealth of knowledge and experiences to us that
open our eyes to new cultures, beliefs, and respect.\"The limits of your language are the limits o
f your world.\"-Ludwig Wittgenstein  Our English learner's have a strong support system at home th
at begs for more resources.  Many times our parents are learning to read and speak English along s
ide of their children.  Sometimes this creates barriers for parents to be able to help their child
learn phonetics, letter recognition, and other reading skills.\r\n\r\nBy providing these dvd's and
players, students are able to continue their mastery of the English language even if no one at hom
e is able to assist.  All families with students within the Level 1 proficiency status, will be a
offered to be a part of this program.  These educational videos will be specially chosen by the En
glish Learner Teacher and will be sent home regularly to watch.  The videos are to help the child
develop early reading skills.\r\n\r\nParents that do not have access to a dvd player will have the
opportunity to check out a dvd player to use for the year.  The plan is to use these videos and ed
ucational dvd's for the years to come for other EL students.\r\nnannan
==================================================
The 51 fifth grade students that will cycle through my classroom this year all love learning, at l
east most of the time. At our school, 97.3% of the students receive free or reduced price lunch. O
f the 560 students, 97.3% are minority students. \r\nThe school has a vibrant community that loves
to get together and celebrate. Around Halloween there is a whole school parade to show off the bea
utiful costumes that students wear. On Cinco de Mayo we put on a big festival with crafts made by
the students, dances, and games. At the end of the year the school hosts a carnival to celebrate t
he hard work put in during the school year, with a dunk tank being the most popular activity.My st
udents will use these five brightly colored Hokki stools in place of regular, stationary, 4-legged
chairs. As I will only have a total of ten in the classroom and not enough for each student to hav
e an individual one, they will be used in a variety of ways. During independent reading time they
will be used as special chairs students will each use on occasion. I will utilize them in place of
chairs at my small group tables during math and reading times. The rest of the day they will be us
ed by the students who need the highest amount of movement in their life in order to stay focused
on school.\r\n\r\nWhenever asked what the classroom is missing, my students always say more Hokki
Stools. They can't get their fill of the 5 stools we already have. When the students are sitting i
n group with me on the Hokki Stools, they are always moving, but at the same time doing their
work. Anytime the students get to pick where they can sit, the Hokki Stools are the first to be ta
ken. There are always students who head over to the kidney table to get one of the stools who are
disappointed as there are not enough of them. \r\n\r\nWe ask a lot of students to sit for 7 hours
a day. The Hokki stools will be a compromise that allow my students to do desk work and move at th
e same time. These stools will help students to meet their 60 minutes a day of movement by
allowing them to activate their core muscles for balance while they sit. For many of my students,
these chairs will take away the barrier that exists in schools for a child who can't sit
still.nannan
==================================================
How do you remember your days of school? Was it in a sterile environment with plain walls, rows of
desks, and a teacher in front of the room? A typical day in our room is nothing like that. I work
hard to create a warm inviting themed room for my students look forward to coming to each day.\r\n
\r\nMy class is made up of 28 wonderfully unique boys and girls of mixed races in
Arkansas.\r\nThey attend a Title I school, which means there is a high enough percentage of free a
nd reduced-price lunch to qualify. Our school is an \"open classroom\" concept, which is very uniq
ue as there are no walls separating the classrooms. These 9 and 10 year-old students are very eage
r learners; they are like sponges, absorbing all the information and experiences and keep on wanti
ng more.With these resources such as the comfy red throw pillows and the whimsical nautical hangin
g decor and the blue fish nets, I will be able to help create the mood in our classroom setting to
```

```
be one of a themed nautical environment. Creating a classroom environment is very important in the success in each and every child's education. The nautical photo props will be used with each child as they step foot into our classroom for the first time on Meet the Teacher evening. I'll take pic tures of each child with them, have them developed, and then hung in our classroom ready for their first day of 4th grade.  This kind gesture will set the tone before even the first day of school! The nautical thank you cards will be used throughout the year by the students as they create thank you cards to their team groups.\r\n\r\nYour generous donations will help me to help make our classroom a fun, inviting, learning environment from day one.\r\n\r\nIt costs lost of money out of my own pocket on resources to get our classroom ready. Please consider helping with this project t o make our new school year a very successful one. Thank you!nannan
==================================================
My kindergarten students have varied disabilities ranging from speech and language delays, cogniti ve delays, gross/fine motor delays, to autism. They are eager beavers and always strive to work th eir hardest working past their limitations. \r\n\r\nThe materials we have are the ones I seek out for my students. I teach in a Title I school where most of the students receive free or reduced pr ice lunch.  Despite their disabilities and limitations, my students love coming to school and come eager to learn and explore.Have you ever felt like you had ants in your pants and you needed to gr oove and move as you were in a meeting? This is how my kids feel all the time. The want to be able to move as they learn or so they say.Wobble chairs are the answer and I love then because they dev elop their core, which enhances gross motor and in Turn fine motor skills. \r\nThey also want to l earn through games, my kids don't want to sit and do worksheets. They want to learn to count by ju mping and playing. Physical engagement is the key to our success. The number toss and color and sh ape mats can make that happen. My students will forget they are doing work and just have the fun a 6 year old deserves.nannan
==================================================
The mediocre teacher tells. The good teacher explains. The superior teacher demonstrates. The grea t teacher inspires. -William A. Ward\r\n\r\nMy school has 803 students which is makeup is 97.6% Af rican-American, making up the largest segment of the student body. A typical school in Dallas is m ade up of 23.2% African-American students. Most of the students are on free or reduced lunch. We a ren't receiving doctors, lawyers, or engineers children from rich backgrounds or neighborhoods. As an educator I am inspiring minds of young children and we focus not only on academics but one smar t, effective, efficient, and disciplined students with good character.In our classroom we can util ize the Bluetooth for swift transitions during class. I use a speaker which doesn't amplify the so und enough to receive the message. Due to the volume of my speaker my students can't hear videos o r books clearly and it isn't making the lessons as meaningful. But with the bluetooth speaker my s tudents will be able to hear and I can stop, pause and replay it at any time.\r\nThe cart will all ow me to have more room for storage of things that are needed for the day and has an extra part to it I can use.  The table top chart has all of the letter, words and pictures for students to learn about different letters and it is more accessible.nannan
==================================================
```

In [13]:

```
# https://stackoverflow.com/a/47091490/4084039
import re

def decontracted(phrase):
    # specific
    phrase = re.sub(r"won't", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)

    # general
    phrase = re.sub(r"n\'t", " not", phrase)
    phrase = re.sub(r"\'re", " are", phrase)
    phrase = re.sub(r"\'s", " is", phrase)
    phrase = re.sub(r"\'d", " would", phrase)
    phrase = re.sub(r"\'ll", " will", phrase)
    phrase = re.sub(r"\'t", " not", phrase)
    phrase = re.sub(r"\'ve", " have", phrase)
    phrase = re.sub(r"\'m", " am", phrase)
    return phrase
```

In [14]:

```
sent = decontracted(project_data['essay'].values[20000])
print(sent)
print("="*50)
```

```
My kindergarten students have varied disabilities ranging from speech and language delays, cogniti ve delays, gross/fine motor delays, to autism. They are eager beavers and always strive to work th eir hardest working past their limitations. \r\n\r\nThe materials we have are the ones I seek out for my students. I teach in a Title I school where most of the students receive free or reduced pr ice lunch.  Despite their disabilities and limitations, my students love coming to school and come eager to learn and explore.Have you ever felt like you had ants in your pants and you needed to gr
```

oove and move as you were in a meeting? This is how my kids feel all the time. The want to be able to move as they learn or so they say.Wobble chairs are the answer and I love then because they develop their core, which enhances gross motor and in Turn fine motor skills. \r\nThey also want to learn through games, my kids do not want to sit and do worksheets. They want to learn to count by jumping and playing. Physical engagement is the key to our success. The number toss and color and shape mats can make that happen. My students will forget they are doing work and just have the fun a 6 year old deserves.nannan
==

In [15]:

```
# \r \n \t remove from string python: http://texthandler.com/info/remove-line-breaks-python/
sent = sent.replace('\\r', ' ')
sent = sent.replace('\\"', ' ')
sent = sent.replace('\\n', ' ')
print(sent)
```

My kindergarten students have varied disabilities ranging from speech and language delays, cognitive delays, gross/fine motor delays, to autism. They are eager beavers and always strive to work their hardest working past their limitations. The materials we have are the ones I seek out for my students. I teach in a Title I school where most of the students receive free or reduced price lunch. Despite their disabilities and limitations, my students love coming to school and come eager to learn and explore.Have you ever felt like you had ants in your pants and you needed to groove and move as you were in a meeting? This is how my kids feel all the time. The want to be able to move as they learn or so they say.Wobble chairs are the answer and I love then because they develop their core, which enhances gross motor and in Turn fine motor skills. They also want to learn through games, my kids do not want to sit and do worksheets. They want to learn to count by jumping and playing. Physical engagement is the key to our success. The number toss and color and shape mats can make that happen. My students will forget they are doing work and just have the fun a 6 year old deserves.nannan

In [16]:

```
#remove spacial character: https://stackoverflow.com/a/5843547/4084039
sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
print(sent)
```

My kindergarten students have varied disabilities ranging from speech and language delays cognitive delays gross fine motor delays to autism They are eager beavers and always strive to work their hardest working past their limitations The materials we have are the ones I seek out for my students I teach in a Title I school where most of the students receive free or reduced price lunch Despite their disabilities and limitations my students love coming to school and come eager to learn and explore Have you ever felt like you had ants in your pants and you needed to groove and move as you were in a meeting This is how my kids feel all the time The want to be able to move as they learn or so they say Wobble chairs are the answer and I love then because they develop their core which enhances gross motor and in Turn fine motor skills They also want to learn through games my kids do not want to sit and do worksheets They want to learn to count by jumping and playing Physical engagement is the key to our success The number toss and color and shape mats can make that happen My students will forget they are doing work and just have the fun a 6 year old deserves nannan

In [17]:

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more',\
```

```
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll'
, 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "d
esn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn',
"mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn',
"wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

In [18]:

```
# Combining all the above stundents
from tqdm import tqdm
preprocessed_essays = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['essay'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e not in stopwords)
    preprocessed_essays.append(sent.lower().strip())
```

```
100%|██████████████████████████████████████████████████████████████████████████████| 109248/109248
[01:44<00:00, 1048.12it/s]
```

In [19]:

```
# after preprocesing
preprocessed_essays[20000]
```

Out[19]:

```
'my kindergarten students varied disabilities ranging speech language delays cognitive delays gros
s fine motor delays autism they eager beavers always strive work hardest working past limitations
the materials ones i seek students i teach title i school students receive free reduced price lunc
h despite disabilities limitations students love coming school come eager learn explore have ever
felt like ants pants needed groove move meeting this kids feel time the want able move learn say w
obble chairs answer i love develop core enhances gross motor turn fine motor skills they also want
learn games kids not want sit worksheets they want learn count jumping playing physical engagement
key success the number toss color shape mats make happen my students forget work fun 6 year old de
serves nannan'
```

In [20]:

```
preprocessed_essays[0]
```

Out[20]:

```
'my students english learners working english second third languages we melting pot refugees immig
rants native born americans bringing gift language school we 24 languages represented english lear
ner program students every level mastery we also 40 countries represented families within school e
ach student brings wealth knowledge experiences us open eyes new cultures beliefs respect the limi
ts language limits world ludwig wittgenstein our english learner strong support system home begs r
esources many times parents learning read speak english along side children sometimes creates
barriers parents able help child learn phonetics letter recognition reading skills by providing dv
d players students able continue mastery english language even no one home able assist all familie
s students within level 1 proficiency status offered part program these educational videos
specially chosen english learner teacher sent home regularly watch the videos help child develop e
arly reading skills parents not access dvd player opportunity check dvd player use year the plan u
se videos educational dvd years come el students nannan'
```

1.7 Preprocessing of `project_title`

In [21]:

```
project_data.head(2)
```

Out[21]:

	Unnamed: 0	id	teacher_id	school_state	project_submitted_datetime	project_title	projec
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	IN	2016-12-05 13:43:57	Educational Support for English Learners at Home	My stu Englisl that ar
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	FL	2016-10-25 09:22:10	Wanted: Projector for Hungry Learners	Our st arrive school lea...

In [22]:

```
# printing some random essays.
print(project_data['project_title'].values[0])
print("="*50)
print(project_data['project_title'].values[150])
print("="*50)
print(project_data['project_title'].values[1000])
print("="*50)
print(project_data['project_title'].values[20000])
print("="*50)
print(project_data['project_title'].values[99999])
print("="*50)
```

```
Educational Support for English Learners at Home
==================================================
More Movement with Hokki Stools
==================================================
Sailing Into a Super 4th Grade Year
==================================================
We Need To Move It While We Input It!
==================================================
Inspiring Minds by Enhancing the Educational Experience
==================================================
```

In [23]:

```
sent_title = decontracted(project_data['project_title'].values[20000])
print(sent_title)
print("="*50)
```

```
We Need To Move It While We Input It!
==================================================
```

In [24]:

```
# \r \n \t remove from string python: http://texthandler.com/info/remove-line-breaks-python/
sent_title = sent_title.replace('\\r', ' ')
sent_title = sent_title.replace('\\"', ' ')
sent_title = sent_title.replace('\\n', ' ')
print(sent_title)
```

```
We Need To Move It While We Input It!
```

In [25]:

```
#remove spacial character: https://stackoverflow.com/a/5843547/4084039
sent_title = re.sub('[^A-Za-z0-9]+', ' ', sent_title)
print(sent_title)
```

```
We Need To Move It While We Input It
```

In [26]:

```
# Combining all the above statemennts
from tqdm import tqdm
preprocessed_title = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['project_title'].values):
    sent_title = decontracted(sentance)
    sent_title = sent_title.replace('\\r', ' ')
    sent_title = sent_title.replace('\\"', ' ')
    sent_title = sent_title.replace('\\n', ' ')
    sent_title = re.sub('[^A-Za-z0-9]+', ' ', sent_title)
    # https://gist.github.com/sebleier/554280
    sent_title = ' '.join(e for e in sent_title.split() if e not in stopwords)
    preprocessed_title.append(sent_title.lower().strip())
```

```
100%|██████████████████████████████████████████████████████████████████████████| 109248/109248
[00:04<00:00, 23975.51it/s]
```

In [27]:

```
# after preprocesing
preprocessed_title[10]
```

Out[27]:

```
'reading changes lives'
```

In [28]:

```
# Combining all the above statemennts
from tqdm import tqdm
preprocessed_prj_sum = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['project_resource_summary'].values):
    sent_title = decontracted(sentance)
    sent_title = sent_title.replace('\\r', ' ')
    sent_title = sent_title.replace('\\"', ' ')
    sent_title = sent_title.replace('\\n', ' ')
    sent_title = re.sub('[^A-Za-z0-9]+', ' ', sent_title)
    # https://gist.github.com/sebleier/554280
    sent_title = ' '.join(e for e in sent_title.split() if e not in stopwords)
    preprocessed_prj_sum.append(sent_title.lower().strip())
```

```
100%|██████████████████████████████████████████████████████████████████████████| 109248/109248
[00:10<00:00, 10086.23it/s]
```

1.8 Numeric feature for Text

1.8.1 Numerric feature for essay

In [29]:

```
# Suggestion 5.you can try improving the score using feature engineering hacks.Try including lengt
h,summary
# and observe the results and re-submit the assignment.

# https://stackoverflow.com/questions/18827198/python-count-number-of-words-in-a-list-strings
preprocessed_essays_wc = []
for item in tqdm(preprocessed_essays):
    preprocessed_essays_wc.append(len(item.split()))
```

```
print(preprocessed_essays_wc[101])
```

```
100%|██████████████████████████████████████████████████████████████| 109248/109248
[00:02<00:00, 53825.08it/s]
```

```
141
```

1.8.2 Numerric feature for title

In [30]:

```
# Suggestion 5.you can try improving the score using feature engineering hacks.Try including lengt
h,summary
# and observe the results and re-submit the assignment.

# https://stackoverflow.com/questions/18827198/python-count-number-of-words-in-a-list-strings
preprocessed_title_wc = []
for item in tqdm(preprocessed_title):
    preprocessed_title_wc.append(len(item.split()))

print(preprocessed_title_wc[101])
```

```
100%|██████████████████████████████████████████████████████████████| 109248/109248
[00:00<00:00, 707461.57it/s]
```

```
3
```

1.8.3 Vectorizing Numerical features

In [31]:

```
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_index()
project_data = pd.merge(project_data, price_data, on='id', how='left')
```

In [32]:

```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-
learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
from sklearn.preprocessing import StandardScaler

# price_standardized = standardScalar.fit(project_data['price'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 329.   ... 399.   287.
73   5.5 ].
# Reshape your data either using array.reshape(-1, 1)

price_scalar = StandardScaler()
price_scalar.fit(project_data['price'].values.reshape(-1,1)) # finding the mean and standard
deviation of this data
print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_scalar.var_[0])}")

# Now standardize the data with above maen and variance.
price_standardized = price_scalar.transform(project_data['price'].values.reshape(-1, 1))
```

```
Mean : 298.1193425966608, Standard deviation : 367.49634838483496
```

In [33]:

```
price_standardized
```

Out[33]:

```
array([[-0.3905327 ],
       [ 0.00239637],
       [ 0.59519138],
       ...,
       [-0.15825829],
```

```
        [-0.61243967],
        [-0.51216657]])
```

Computing Sentiment Scores

In [34]:

```
## https://monkeylearn.com/sentiment-analysis/
## http://t-redactyl.io/blog/2017/04/using-vader-to-handle-sentiment-analysis-with-social-media-te
xt.html
#
#import nltk
#from nltk.sentiment.vader import SentimentIntensityAnalyzer
#
#import nltk
#nltk.download('vader_lexicon')
#
#sid = SentimentIntensityAnalyzer()
#
#for_sentiment = 'a person is a person no matter how small dr seuss i teach the smallest students
with the biggest enthusiasm \
#for learning my students learn in many different ways using all of our senses and multiple intell
igences i use a wide range\
#of techniques to help all my students succeed students in my class come from a variety of differe
nt backgrounds which makes\
#for wonderful sharing of experiences and cultures including native americans our school is a cari
ng community of successful \
#learners which can be seen through collaborative student project based learning in and out of the
classroom kindergarteners \
#in my class love to work with hands on materials and have many different opportunities to
practice a skill before it is\
#mastered having the social skills to work cooperatively with friends is a crucial aspect of the k
indergarten curriculum\
#montana is the perfect place to learn about agriculture and nutrition my students love to role pl
ay in our pretend kitchen\
#in the early childhood classroom i have had several kids ask me can we try cooking with real food
i will take their idea \
#and create common core cooking lessons where we learn important math and writing concepts while c
ooking delicious healthy \
#food for snack time my students will have a grounded appreciation for the work that went into mak
ing the food and knowledge \
#of where the ingredients came from as well as how it is healthy for their bodies this project wou
ld expand our learning of \
#nutrition and agricultural cooking recipes by having us peel our own apples to make homemade appl
esauce make our own bread \
#and mix up healthy plants from our classroom garden in the spring we will also create our own coo
kbooks to be printed and \
#shared with families students will gain math and literature skills as well as a life long enjoyme
nt for healthy cooking \
#nannan'
#ss = sid.polarity_scores(for_sentiment)
#
## The end=' ' is just to say that you want a space after the end of the statement instead of a ne
w line character.
#for k in ss:
#    print('{0}: {1}, '.format(k, ss[k]), end='')
#
#for k in ss:
#    print('{0}: {1}, '.format(k, ss[k]))
#
# we can use these 4 things as features/attributes (neg, neu, pos, compound)
# neg: 0.0, neu: 0.753, pos: 0.247, compound: 0.93
#print(type(ss))
#print(ss)
```

In [35]:

```
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer

import nltk
nltk.download('vader_lexicon')
```

```
sid = SentimentIntensityAnalyzer()

from tqdm import tqdm
from tqdm import tqdm_notebook
preprocessed_sentiments = []
# tqdm is for printing the status bar
for sentance in tqdm_notebook(project_data['essay'].values):
    sentiment = []
    sentiment = sid.polarity_scores(sentance)
    preprocessed_sentiments.append([sentiment['neg'], sentiment['pos'], sentiment['neu'],
sentiment['compound']])
```

```
C:\Users\samar\Anaconda3\lib\site-packages\nltk\twitter\__init__.py:20: UserWarning:

The twython library has not been installed. Some functionality from the twitter package will not b
e available.
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     C:\Users\samar\AppData\Roaming\nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
```

In [36]:

```
print(type(preprocessed_sentiments))
print(preprocessed_sentiments[1:5])
#print(preprocessed_sentiments([sentiment['neg']]))
print(sentiment['neg'])

project_data[['neg', 'pos', 'neu', 'compound']] = pd.DataFrame(preprocessed_sentiments)
```

```
<class 'list'>
[[0.037, 0.112, 0.851, 0.9267], [0.058, 0.179, 0.764, 0.995], [0.052, 0.214, 0.733, 0.9931], [0.01
6, 0.087, 0.897, 0.9192]]
0.023
```

In [37]:

```
print(project_data.columns.values)
project_data['neg'].values
```

```
['Unnamed: 0' 'id' 'teacher_id' 'school_state'
 'project_submitted_datetime' 'project_title' 'project_essay_1'
 'project_essay_2' 'project_essay_3' 'project_essay_4'
 'project_resource_summary' 'teacher_number_of_previously_posted_projects'
 'project_is_approved' 'clean_categories' 'clean_subcategories'
 'clean_grade' 'clean_tea_pfx' 'essay' 'price' 'quantity' 'neg' 'pos'
 'neu' 'compound']
```

Out[37]:

```
array([0.008, 0.037, 0.058, ..., 0.   , 0.013, 0.023])
```

Adding word count for essay and Title

In [38]:

```
project_data['essay_wc'] = preprocessed_essays_wc

project_data['title_wc'] = preprocessed_title_wc
```

Adding Preprocessed essay and Preprocessed Title

In [39]:

```
project_data['essay'] = preprocessed_essays
project_data['project_title'] = preprocessed_title
```

```
project_data['project_title'] = preprocessed_title
```

In [40]:

```
project_data.columns
```

Out[40]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'school_state',
       'project_submitted_datetime', 'project_title', 'project_essay_1',
       'project_essay_2', 'project_essay_3', 'project_essay_4',
       'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approved',
       'clean_categories', 'clean_subcategories', 'clean_grade',
       'clean_tea_pfx', 'essay', 'price', 'quantity', 'neg', 'pos', 'neu',
       'compound', 'essay_wc', 'title_wc'],
      dtype='object')
```

1.9 Preparing data for models

In [41]:

```
project_data.columns
```

Out[41]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'school_state',
       'project_submitted_datetime', 'project_title', 'project_essay_1',
       'project_essay_2', 'project_essay_3', 'project_essay_4',
       'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approved',
       'clean_categories', 'clean_subcategories', 'clean_grade',
       'clean_tea_pfx', 'essay', 'price', 'quantity', 'neg', 'pos', 'neu',
       'compound', 'essay_wc', 'title_wc'],
      dtype='object')
```

we are going to consider

```
- school_state : categorical data
- clean_categories : categorical data
- clean_subcategories : categorical data
- project_grade_category : categorical data
- teacher_prefix : categorical data

- project_title : text data
- text : text data
- project_resource_summary: text data (optinal)

- quantity : numerical (optinal)
- teacher_number_of_previously_posted_projects : numerical
- price : numerical
```

Computing Sentiment Scores

In [42]:

```
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer

# import nltk
# nltk.download('vader_lexicon')

sid = SentimentIntensityAnalyzer()

for_sentiment = 'a person is a person no matter how small dr seuss i teach the smallest students w
ith the biggest enthusiasm \
for learning my students learn in many different ways using all of our senses and multiple intelli
gences i use a wide range\
```

```
of techniques to help all my students succeed students in my class come from a variety of differen
t backgrounds which makes\
for wonderful sharing of experiences and cultures including native americans our school is a carin
g community of successful \
learners which can be seen through collaborative student project based learning in and out of the
classroom kindergarteners \
in my class love to work with hands on materials and have many different opportunities to practice
a skill before it is\
mastered having the social skills to work cooperatively with friends is a crucial aspect of the ki
ndergarten curriculum\
montana is the perfect place to learn about agriculture and nutrition my students love to role pla
y in our pretend kitchen\
in the early childhood classroom i have had several kids ask me can we try cooking with real food
i will take their idea \
and create common core cooking lessons where we learn important math and writing concepts while co
oking delicious healthy \
food for snack time my students will have a grounded appreciation for the work that went into maki
ng the food and knowledge \
of where the ingredients came from as well as how it is healthy for their bodies this project woul
d expand our learning of \
nutrition and agricultural cooking recipes by having us peel our own apples to make homemade apple
sauce make our own bread \
and mix up healthy plants from our classroom garden in the spring we will also create our own cook
books to be printed and \
shared with families students will gain math and literature skills as well as a life long enjoymen
t for healthy cooking \
nannan'
ss = sid.polarity_scores(for_sentiment)

for k in ss:
    print('{0}: {1}, '.format(k, ss[k]), end='')

# we can use these 4 things as features/attributes (neg, neu, pos, compound)
# neg: 0.01, neu: 0.745, pos: 0.245, compound: 0.9975
```

```
neg: 0.01, neu: 0.745, pos: 0.245, compound: 0.9975,
```

Assignment 11: TruncatedSVD

- step 1 Select the top 2k words from essay text and project_title (concatinate essay text with project title and then find the top 2k words) based on their `idf_` values
- step 2 Compute the co-occurance matrix with these 2k words, with window size=5 (ref)
- step 3 Use TruncatedSVD on calculated co-occurance matrix and reduce its dimensions, choose the number of components (`n_components`) using elbow method

  > - The shape of the matrix after TruncatedSVD will be 2000*n, i.e. each row represents a vector form of the corresponding word.
  > - Vectorize the essay text and project titles using these word vectors. (while vectorizing, do ignore all the words which are not in top 2k words)

- step 4 Concatenate these truncatedSVD matrix, with the matrix with features
  - **school_state** : categorical data
  - **clean_categories** : categorical data
  - **clean_subcategories** : categorical data
  - **project_grade_category** :categorical data
  - **teacher_prefix** : categorical data
  - **quantity** : numerical data
  - **teacher_number_of_previously_posted_projects** : numerical data
  - **price** : numerical data
  - **sentiment score's of each of the essay** : numerical data
  - **number of words in the title** : numerical data
  - **number of words in the combine essays** : numerical data
  - **word vectors calculated in** step 3 : numerical data
- step 5: Apply GBDT on matrix that was formed in step 4 of this assignment, **DO REFER THIS BLOG: XGBOOST DMATRIX**
- **step 6:Hyper parameter tuning (Consider any two hyper parameters)**
  - **Find the best hyper parameter which will give the maximum AUC value**
  - **Find the best hyper paramter using k-fold cross validation or simple cross validation data**

- **Use gridsearch cv or randomsearch cv or you can also write your own for loops to do this task of hyperparameter tuning**

In [43]:

```
##taking 50K datapoint
project_data50K=project_data[:50000]
#project_data100K=project_data[:100000]
#X=project_data100K
X=project_data50K
print(project_data50K.shape)
#print(project_data100K.shape)
print(X.shape)
```

```
(50000, 26)
(50000, 26)
```

In [44]:

```
y = project_data['project_is_approved'].values
project_data.drop(['project_is_approved'], axis=1, inplace=True)
#print(y.shape)
project_data.head(1)

y50K=y[:50000]
y=y50K
```

In [45]:

```
print(X.shape)
print(y.shape)
```

```
(50000, 26)
(50000,)
```

In [46]:

```
# train test split | https://scikit-
learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
# spliting Xq and Yq in Train(further into Train and CV) and Test matrix
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, stratify=y)
#X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=0.33,
stratify=y_train)

print(X_train.shape, y_train.shape)
#print(X_cv.shape, y_cv.shape)
print(X_test.shape, y_test.shape)

print("="*100)
```

```
(33500, 26) (33500,)
(16500, 26) (16500,)
====================================================================================================
```

2.1.1 Make Data Model Ready: encoding school_state categorical data

In [47]:

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(min_df=10,ngram_range=(1,2), max_features=5000)
vectorizer.fit(X_train['school_state'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_state_ohe = vectorizer.transform(X_train['school_state'].values)
#X_cv_state_ohe = vectorizer.transform(X_cv['school_state'].values)
X_test_state_ohe = vectorizer.transform(X_test['school_state'].values)
```

```
print("school_state After vectorizations")
print(X_train_state_ohe.shape, y_train.shape)
#print(X_cv_state_ohe.shape, y_cv.shape)
print(X_test_state_ohe.shape, y_test.shape)
st=vectorizer.get_feature_names()
print(vectorizer.get_feature_names())
print("="*100)
```

```
school_state After vectorizations
(33500, 51) (33500,)
(16500, 51) (16500,)
['ak', 'al', 'ar', 'az', 'ca', 'co', 'ct', 'dc', 'de', 'fl', 'ga', 'hi', 'ia', 'id', 'il', 'in', 'k
s', 'ky', 'la', 'ma', 'md', 'me', 'mi', 'mn', 'mo', 'ms', 'mt', 'nc', 'nd', 'ne', 'nh', 'nj', 'nm',
'nv', 'ny', 'oh', 'ok', 'or', 'pa', 'ri', 'sc', 'sd', 'tn', 'tx', 'ut', 'va', 'vt', 'wa', 'wi', 'wv
', 'wy']
====================================================================================================
```

2.1.2 Make Data Model Ready: encoding clean_categories

In [48]:

```
from sklearn.feature_extraction.text import CountVectorizer
#vectorizer = CountVectorizer(min_df=10,ngram_range=(1,2), max_features=5000)
vectorizer = CountVectorizer(vocabulary =list(sorted_cat_dict.keys()),lowercase =False,binary=True
)
vectorizer.fit(X_train['clean_categories'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_clean_ohe = vectorizer.transform(X_train['clean_categories'].values)
#X_cv_clean_ohe = vectorizer.transform(X_cv['clean_categories'].values)
X_test_clean_ohe = vectorizer.transform(X_test['clean_categories'].values)

print("clean_categories After vectorizations")
print(X_train_clean_ohe.shape, y_train.shape)
#print(X_cv_clean_ohe.shape, y_cv.shape)
print(X_test_clean_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
cc=vectorizer.get_feature_names()
print(cc)
print("="*100)
```

```
clean_categories After vectorizations
(33500, 9) (33500,)
(16500, 9) (16500,)
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds',
'Health_Sports', 'Math_Science', 'Literacy_Language']
====================================================================================================

['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds',
'Health_Sports', 'Math_Science', 'Literacy_Language']
====================================================================================================
```

2.1.3 Make Data Model Ready: encoding clean_subcategories

In [49]:

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(vocabulary =list(sorted_sub_cat_dict.keys()),lowercase =False,binary=
True)
vectorizer.fit(X_train['clean_subcategories'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_cleanSub_ohe = vectorizer.transform(X_train['clean_subcategories'].values)
#X_cv_cleanSub_ohe = vectorizer.transform(X_cv['clean_subcategories'].values)
X_test_cleanSub_ohe = vectorizer.transform(X_test['clean_subcategories'].values)

print("clean_subcategories After vectorizations")
print(X_train_cleanSub_ohe.shape, y_train.shape)
```

```
#print(X_cv_cleanSub_ohe.shape, y_cv.shape)
print(X_test_cleanSub_ohe.shape, y_test.shape)
cst=vectorizer.get_feature_names()
#print(cst)
print("="*100)
```

```
clean_subcategories After vectorizations
(33500, 30) (33500,)
(16500, 30) (16500,)
==========================================================================================================
```

2.1.4 Make Data Model Ready: encoding project_grade_category

In [50]:

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(vocabulary =list(sorted_prj_grade_cat_dict.keys()),lowercase =False,b
inary=True)
vectorizer.fit(X_train['clean_grade'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_grade_ohe = vectorizer.transform(X_train['clean_grade'].values)
#X_cv_grade_ohe = vectorizer.transform(X_cv['clean_grade'].values)
X_test_grade_ohe = vectorizer.transform(X_test['clean_grade'].values)

print("project_grade_category After vectorizations")
print(X_train_grade_ohe.shape, y_train.shape)
#print(X_cv_grade_ohe.shape, y_cv.shape)
print(X_test_grade_ohe.shape, y_test.shape)

pgc=vectorizer.get_feature_names()
print(pgc)
print("="*100)
```

```
project_grade_category After vectorizations
(33500, 4) (33500,)
(16500, 4) (16500,)
['9-12', '6-8', '3-5', 'PreK-2']
==========================================================================================================
```

2.1.5 Make Data Model Ready: encoding teacher_prefix

In [51]:

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(vocabulary =list(sorted_tea_pfx_cat_dict.keys()),lowercase =False,bin
ary=True)
#https://stackoverflow.com/questions/52736900/how-to-solve-the-attribute-error-float-object-has-no
-attribute-split-in-pyth
vectorizer.fit(X_train['clean_tea_pfx'].astype(str).values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_teacher_ohe = vectorizer.transform(X_train['clean_tea_pfx'].astype(str).values)
#X_cv_teacher_ohe = vectorizer.transform(X_cv['clean_tea_pfx'].astype(str).values)
X_test_teacher_ohe = vectorizer.transform(X_test['clean_tea_pfx'].astype(str).values)

print("teacher_prefix After vectorizations")
print(X_train_teacher_ohe.shape, y_train.shape)
#print(X_cv_teacher_ohe.shape, y_cv.shape)
print(X_test_teacher_ohe.shape, y_test.shape)
tp=vectorizer.get_feature_names()
print(tp)
print("="*100)
```

```
teacher_prefix After vectorizations
(33500, 5) (33500,)
(16500, 5) (16500,)
['Dr', 'Teacher', 'Mr', 'Ms', 'Mrs']
```

```
==========================================================================================================
```

2.2 Make Data Model Ready: encoding numerical, categorical features

2.2.1 Make Data Model Ready: encoding numerical | quantity

In [52]:

```
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
# normalizer.fit(X['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
normalizer.fit(X_train['quantity'].values.reshape(-1,1))

X_train_quantity_norm = normalizer.transform(X_train['quantity'].values.reshape(-1,1))
#X_cv_quantity_norm = normalizer.transform(X_cv['quantity'].values.reshape(-1,1))
X_test_quantity_norm = normalizer.transform(X_test['quantity'].values.reshape(-1,1))

print("quantity After vectorizations")
print(X_train_quantity_norm.shape, y_train.shape)
#print(X_cv_quantity_norm.shape, y_cv.shape)
print(X_test_quantity_norm.shape, y_test.shape)
print("="*100)
```

```
quantity After vectorizations
(33500, 1) (33500,)
(16500, 1) (16500,)
==========================================================================================================
```

2.2.2 Make Data Model Ready: encoding numerical| teacher_number_of_previously_posted_projects

In [53]:

```
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
# normalizer.fit(X['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
normalizer.fit(X_train['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))

X_train_TprevPrj_norm =
normalizer.transform(X_train['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))
#X_cv_TprevPrj_norm =
normalizer.transform(X_cv['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))
X_test_TprevPrj_norm = normalizer.transform(X_test['teacher_number_of_previously_posted_projects']
.values.reshape(-1,1))

print("teacher_number_of_previously_posted_projects After vectorizations")
print(X_train_TprevPrj_norm.shape, y_train.shape)
#print(X_cv_TprevPrj_norm.shape, y_cv.shape)
print(X_test_TprevPrj_norm.shape, y_test.shape)
print("="*100)
```

```
teacher_number_of_previously_posted_projects After vectorizations
(33500, 1) (33500,)
(16500, 1) (16500,)
==========================================================================================================
```

2.2.3 Make Data Model Ready: encoding numerical | price

In [54]:

```
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
# normalizer.fit(X['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
normalizer.fit(X_train['price'].values.reshape(-1,1))

X_train_price_norm = normalizer.transform(X_train['price'].values.reshape(-1,1))
#X_cv_price_norm = normalizer.transform(X_cv['price'].values.reshape(-1,1))
X_test_price_norm = normalizer.transform(X_test['price'].values.reshape(-1,1))

print("Price After vectorizations")
print(X_train_price_norm.shape, y_train.shape)
#print(X_cv_price_norm.shape, y_cv.shape)
print(X_test_price_norm.shape, y_test.shape)
print("="*100)
```

```
Price After vectorizations
(33500, 1) (33500,)
(16500, 1) (16500,)
====================================================================================================
```

In [55]:

```
h=['price','quantity','teacher_number_of_previously_posted_projects']
print(type(h))
```

```
<class 'list'>
```

2.2.4 Make Data Model Ready: encoding numerical | sentimental score

2.2.4.1 Make Data Model Ready: encoding numerical | sentimental score | neg

In [56]:

```
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
# normalizer.fit(X['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
normalizer.fit(X_train['neg'].values.reshape(-1,1))

X_train_neg_norm = normalizer.transform(X_train['neg'].values.reshape(-1,1))
#X_cv_neg_norm = normalizer.transform(X_cv['neg'].values.reshape(-1,1))
X_test_neg_norm = normalizer.transform(X_test['neg'].values.reshape(-1,1))

print("neg After vectorizations")
print(X_train_neg_norm.shape, y_train.shape)
#print(X_cv_neg_norm.shape, y_cv.shape)
print(X_test_neg_norm.shape, y_test.shape)
print("="*100)
```

```
neg After vectorizations
(33500, 1) (33500,)
(16500, 1) (16500,)
```

```
==========================================================================================================
```

2.2.4.2 Make Data Model Ready: encoding numerical | sentimental score | pos

In [57]:

```
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
# normalizer.fit(X['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
normalizer.fit(X_train['pos'].values.reshape(-1,1))

X_train_pos_norm = normalizer.transform(X_train['pos'].values.reshape(-1,1))
#X_cv_pos_norm = normalizer.transform(X_cv['pos'].values.reshape(-1,1))
X_test_pos_norm = normalizer.transform(X_test['pos'].values.reshape(-1,1))

print("pos After vectorizations")
print(X_train_pos_norm.shape, y_train.shape)
#print(X_cv_pos_norm.shape, y_cv.shape)
print(X_test_pos_norm.shape, y_test.shape)
print("="*100)
```

```
pos After vectorizations
(33500, 1) (33500,)
(16500, 1) (16500,)
==========================================================================================================
```

2.2.4.3 Make Data Model Ready: encoding numerical | sentimental score | neu

In [58]:

```
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
# normalizer.fit(X['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
normalizer.fit(X_train['neu'].values.reshape(-1,1))

X_train_neu_norm = normalizer.transform(X_train['neu'].values.reshape(-1,1))
#X_cv_neu_norm = normalizer.transform(X_cv['neu'].values.reshape(-1,1))
X_test_neu_norm = normalizer.transform(X_test['neu'].values.reshape(-1,1))

print("neu After vectorizations")
print(X_train_neu_norm.shape, y_train.shape)
#print(X_cv_neu_norm.shape, y_cv.shape)
print(X_test_neu_norm.shape, y_test.shape)
print("="*100)
```

```
neu After vectorizations
(33500, 1) (33500,)
(16500, 1) (16500,)
==========================================================================================================
```

2.2.4.4 Make Data Model Ready: encoding numerical | sentimental score | compound

In [59]:

```
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
```

```
# normalizer.fit(X['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
normalizer.fit(X_train['compound'].values.reshape(-1,1))

X_train_compound_norm = normalizer.transform(X_train['compound'].values.reshape(-1,1))
#X_cv_compound_norm = normalizer.transform(X_cv['compound'].values.reshape(-1,1))
X_test_compound_norm = normalizer.transform(X_test['compound'].values.reshape(-1,1))

print("compound After vectorizations")
print(X_train_compound_norm.shape, y_train.shape)
#print(X_cv_compound_norm.shape, y_cv.shape)
print(X_test_compound_norm.shape, y_test.shape)
print("="*100)
```

```
compound After vectorizations
(33500, 1) (33500,)
(16500, 1) (16500,)
====================================================================================================
```

2.2.5 Make Data Model Ready: encoding numerical | number of words in the title

In [60]:

```
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
# normalizer.fit(X['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
normalizer.fit(X_train['title_wc'].values.reshape(-1,1))

X_train_title_wc_norm = normalizer.transform(X_train['title_wc'].values.reshape(-1,1))
#X_cv_title_wc_norm = normalizer.transform(X_cv['title_wc'].values.reshape(-1,1))
X_test_title_wc_norm = normalizer.transform(X_test['title_wc'].values.reshape(-1,1))

print("title_wc After vectorizations")
print(X_train_title_wc_norm.shape, y_train.shape)
#print(X_cv_title_wc_norm.shape, y_cv.shape)
print(X_test_title_wc_norm.shape, y_test.shape)
print("="*100)
```

```
title_wc After vectorizations
(33500, 1) (33500,)
(16500, 1) (16500,)
====================================================================================================
```

2.2.6 Make Data Model Ready: encoding numerical | number of words in the essay

In [61]:

```
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
# normalizer.fit(X['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
normalizer.fit(X_train['essay_wc'].values.reshape(-1,1))
```

```
X_train_essay_wc_norm = normalizer.transform(X_train['essay_wc'].values.reshape(-1,1))
#X_cv_essay_wc_norm = normalizer.transform(X_cv['essay_wc'].values.reshape(-1,1))
X_test_essay_wc_norm = normalizer.transform(X_test['essay_wc'].values.reshape(-1,1))

print("essay_wc After vectorizations")
print(X_train_essay_wc_norm.shape, y_train.shape)
#print(X_cv_essay_wc_norm.shape, y_cv.shape)
print(X_test_essay_wc_norm.shape, y_test.shape)
print("="*100)
```

```
essay_wc After vectorizations
(33500, 1) (33500,)
(16500, 1) (16500,)
====================================================================================================
```

2. TruncatedSVD

2.1 Selecting top 2000 words from `essay` and `project_title`

In [62]:

```
# please write all the code with proper documentation, and proper titles for each subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your code
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

- **step 1 Select the top 2k words from essay text and project_title (concatinate essay text with project title and then find the top 2k words) based on their `idf_` values**

In [63]:

```
# https://stackoverflow.com/questions/19377969/combine-two-columns-of-text-in-dataframe-in-pandas-python
# dataframe["period"] = dataframe["Year"].map(str) + dataframe["quarter"]
#project_data.info()
#print("Essay")
#print(project_data.essay[0])
#print(project_data.project_title[0])
#print(project_data.essay.head(2))
#print("Project_title")
#print(project_data.project_title.head(2))
X_train["EssayTitle"] = X_train.essay + X_train.project_title
X_test["EssayTitle"] = X_test.essay + X_test.project_title

#X_train["EssayTitle"] = X_train.preprocessed_essays+X_train.preprocessed_title
#X_test["EssayTitle"] = X_test.preprocessed_essays+X_test.preprocessed_title
#print(project_data.columns)
print(X_train.columns)
print(X_train.shape)
#print("EssayTitle")
#print(project_data.EssayTitle[0])
```

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'school_state',
       'project_submitted_datetime', 'project_title', 'project_essay_1',
       'project_essay_2', 'project_essay_3', 'project_essay_4',
       'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approved',
       'clean_categories', 'clean_subcategories', 'clean_grade',
       'clean_tea_pfx', 'essay', 'price', 'quantity', 'neg', 'pos', 'neu',
       'compound', 'essay_wc', 'title_wc', 'EssayTitle'],
      dtype='object')
(33500, 27)
```

In [64]:

```
#X_train["EssayTitle"]
#
#9115     liberty elementary title 1 school large percen...
#13389    my students come diverse backgrounds they come...
#13827    environment shapes experience no less true cla...
#7263     my students diverse group ambitious enthusiast...
#45303    my students diverse ethnicity also abilities t...
#30987    my students love coming school everyday i teac...
#16803    music truly helps change lives better for stud...
#37341    our school consists entirely k 5 students spec...
#37610    i 23 first graders nine girls fourteen boys th...
#45753    my students enthusiastic dynamic resourceful l...#
```

In [65]:

```
# https://scikit-
learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html
#idf_ : array, shape (n_features)
#The inverse document frequency (IDF) vector; only defined if use_idf is True.

from sklearn.feature_extraction.text import TfidfVectorizer

Tfidf_vectorizer = TfidfVectorizer(min_df=10,ngram_range=(1,1), max_features=5000,use_idf=True)
X_text_tfidf = Tfidf_vectorizer.fit_transform(X_train['EssayTitle'].values)

print("Essay After vectorizations")
tf=Tfidf_vectorizer.get_feature_names()
#print(tf)
print("="*100)



#Essay After vectorizations
#['00', '000', '10', '100', '1000', '10th', '11', '110', '11th', '12', '120', '12th', '13', '14',
'15', '150', '16',
#'17', '18', '180', '19', '1st', '20', '200', '2015', '2016', '2017', '21', '21st', '22', '23', '2
4', '25', '26', '27',
#'28', '29', '2nd', '30', '300', '31', '32', '33', '34', '35', '36', '3d', '3doodler', '3rd',
'40', '400', '45', '450',
#'48', '4th', '50', '500', '55', '5th', '60', '600', '65', '6th', '70', '700', '74', '75', '7th',
'80', '800', '84', '85',
#'8th', '90', '900', '92', '94', '95', '96', '97', '98', '99', '9th', 'abc', 'abilities',
'ability', 'able', 'about',
#'absent', 'absolute', 'absolutely', 'absorb', 'abstract', 'abundance', 'abuse', 'academic', 'acad
emically', 'academics',
#'academy', 'accelerated', 'accept', 'acceptance', 'accepted', 'accepting', 'access', 'accessed',
'accessibility',
#'accessible', 'accessing', 'accessories', 'accidents', 'accommodate', 'accommodations',
'accompany', 'accomplish',
#'accomplished', 'accomplishing', 'accomplishment', 'accomplishments', 'according', 'account', 'ac
countability',
#'accountable', 'accounts', 'accuracy', 'accurate', 'accurately', 'accustomed', 'achieve', 'achiev
ed', 'achievement',
#'achievements', 'achievers', 'achieving', 'acquire', 'acquired', 'acquiring', 'acquisition', 'acr
oss', 'act', 'acting', 'action', 'actions', 'activate', 'active', 'actively', 'activities', 'activ
it
```

```
Essay After vectorizations
====================================================================================================
```

IDF and wrod (Features name) for AVG W2V

Taking words, so later we can use it for deriving vectorize of both essay and title.

In [66]:

```
print(Tfidf_vectorizer.idf_)

df_idf = pd.DataFrame(Tfidf_vectorizer.idf_, index=tf,columns=["tf_idf_weights"])
df_idf.sort_desc=df_idf.sort_values(by=["tf_idf_weights"],ascending=False)
```

```
#df_idf_sort_desc=df_idf_sort_desc
df_idf_sort_desc_2k=df_idf_sort_desc[:2000]
df_idf_sort_desc_2k
```

```
[7.34179312 6.01665319 4.5185999  ... 6.90847106 6.30133676 7.21463795]
```

Out[66]:

	tf_idf_weights
archery	8.474892
dell	8.200455
smoothies	8.200455
lacrosse	8.087126
runners	8.052035
swim	7.985343
hockey	7.985343
chicken	7.985343
chickens	7.985343
golf	7.953595
bot	7.922823
oils	7.892970
violin	7.892970
guitars	7.863983
echo	7.835812
minecraft	7.835812
calculus	7.808413
whisper	7.808413
origami	7.808413
volleyballs	7.808413
easels	7.781744
drone	7.781744
fans	7.781744
compass	7.781744
waves	7.781744
printmaking	7.781744
birthday	7.755769
civilizations	7.755769
camp	7.755769
sewing	7.755769
...	...
squares	6.559518
expectation	6.559518
melting	6.559518
equally	6.559518
transitioning	6.559518
disadvantages	6.559518
statuses	6.559518

	tf_idf_weights
silent	6.559518
modalities	6.559518
flash	6.559518
suggested	6.559518
supplemental	6.559518
luxury	6.551796
illustrate	6.551796
arizona	6.551796
700	6.551796
passions	6.551796
avoid	6.551796
nannannew	6.551796
introduction	6.551796
homelessness	6.551796
regulation	6.551796
hoops	6.551796
window	6.551796
acquired	6.551796
appeal	6.551796
kinders	6.551796
breaking	6.551796
fairly	6.551796
kept	6.544133

2000 rows × 1 columns

In [67]:

```
print(type(Tfidf_vectorizer.idf_))
print(Tfidf_vectorizer.idf_)

# argsort() will return the indices of values from low to high.
# When you print feature names of these indices, these indices will return you the feature names w
ith low probability.
# So, please reverse the indices after argsort()

tf_sorted_Asc=Tfidf_vectorizer.idf_.argsort()
print(tf_sorted_Asc)
tf_sorted_desc=tf_sorted_Asc[::-1]
print(tf_sorted_desc)

# https://cmdlinetips.com/2018/01/how-to-create-pandas-dataframe-from-multiple-lists/
TFIDF_Feature_IDX_dataFrame=pd.DataFrame({'Feature_Word': tf,'Feature_index' : tf_sorted_desc})

# https://cmdlinetips.#com/2018/02/how-to-sort-pandas-dataframe-by-columns-and-row/
TFIDF_Feature_IDX_dataFrame_sorted=TFIDF_Feature_IDX_dataFrame.sort_values('Feature_index',ascendir
g=False)
TFIDF_Feature_IDX_dataFrame_sorted.head(11)

TFIDF_Feature_2K=TFIDF_Feature_IDX_dataFrame_sorted[:2000]
print(type(TFIDF_Feature_2K))
TFIDF_Feature_2K

#TFIDF_Feature_40=TFIDF_Feature_IDX_dataFrame_sorted[:40]
#print(type(TFIDF_Feature_40))
#TFIDF_Feature_40
```

```
<class 'numpy.ndarray'>
[7.34179312 6.01665319 4.5185999  ... 6.90847106 6.30133676 7.21463795]
[4357 3959 2943 ... 1169 4146  315]
```

```
[ 315 4146 1169 ... 2943 3959 4357]
<class 'pandas.core.frame.DataFrame'>
```

Out[67]:

	Feature_Word	Feature_index
677	cannot	4999
2392	intelligence	4998
1333	dollars	4997
790	christmas	4996
1225	device	4995
2827	microscope	4994
4616	toward	4993
2185	home	4992
3638	questioners	4991
4798	virginia	4990
4485	teen	4989
3556	production	4988
3991	security	4987
4612	touches	4986
3050	nations	4985
4858	weather	4984
513	benjamin	4983
1227	devoted	4982
334	articles	4981
360	assignment	4980
4975	wrote	4979
717	causing	4978
64	70	4977
390	audio	4976
1549	enough	4975
2164	highlight	4974
4023	sequence	4973
285	anytime	4972
4864	weekends	4971
4198	south	4970
...	...	...
1001	continue	3029
1763	fellow	3028
964	connected	3027
2885	monday	3026
808	classic	3025
714	cause	3024
976	consist	3023
1186	deployed	3022
2132	headsets	3021

	Feature_Word	Feature_index
1384	drumming	3020
3564	proficient	3019
2225	hugs	3018
787	choosing	3017
500	believing	3016
2171	historic	3015
372	athletes	3014
661	calculators	3013
765	charts	3012
410	aware	3011
1982	genius	3010
1695	external	3009
492	behind	3008
852	codes	3007
1943	funding	3006
1022	cooperatively	3005
2887	monitor	3004
3030	nannansupplies	3003
1597	essentials	3002
1177	demonstrated	3001
608	bring	3000

2000 rows × 2 columns

In [68]:

```
#TFIDF_Feature_40=TFIDF_Feature_IDX_dataFrame_sorted[:40]
#print(type(TFIDF_Feature_40))
#TFIDF_Feature_40
#
#TFIDF_Feature_EssTitle_40=TFIDF_Feature_40.drop(columns="Feature_index")
#print(type(TFIDF_Feature_EssTitle_40))
#print(TFIDF_Feature_EssTitle_40)
```

In [69]:

```
TFIDF_Feature_EssTitle=TFIDF_Feature_2K.drop(columns="Feature_index")
print(type(TFIDF_Feature_EssTitle))
print(TFIDF_Feature_EssTitle)
```

```
<class 'pandas.core.frame.DataFrame'>
        Feature_Word
677           cannot
2392    intelligence
1333         dollars
790        christmas
1225          device
2827      microscope
4616          toward
2185            home
3638     questioners
4798        virginia
4485            teen
3556      production
3991        security
4612         touches
3050         nations
4858         weather
513          benjamin
1227          devoted
```

```
334          articles
360        assignment
4975            wrote
717           causing
64                 70
390             audio
1549           enough
2164        highlight
4023         sequence
285           anytime
4864         weekends
4198            south
...               ...
1001         continue
1763           fellow
964         connected
2885           monday
808           classic
714             cause
976           consist
1186         deployed
2132         headsets
1384         drumming
3564       proficient
2225             hugs
787          choosing
500         believing
2171         historic
372          athletes
661       calculators
765            charts
410             aware
1982           genius
1695         external
492            behind
852             codes
1943          funding
1022   cooperatively
2887          monitor
3030  nannansupplies
1597      essentials
1177    demonstrated
608             bring

[2000 rows x 1 columns]
```

In [70]:

```
#TFIDF_Feature_EssTitle.Feature_Word.value_counts()
```

2.2 Computing Co-occurance matrix

In [71]:

```
# please write all the code with proper documentation, and proper titles for each subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your code
# make sure you featurize train and test data separatly

# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

I tried to run first example mentioend in below link

https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/

count-word2veec/

In [72]:

```
# https://stackoverflow.com/questions/41661801/python-calculate-the-co-occurrence-matrix
length=6
CoMatrix = np.zeros([length,length]) # n is the count of all words

print(CoMatrix[1,2])
print(CoMatrix.shape)
print(type(CoMatrix))
#def cal_occ(TFIDF_EssTitle_npAarr, listofSentence,CoMatrix):
def cal_occ(CoMatDF,CorpusList):
    # https://www.geeksforgeeks.org/enumerate-in-python/
    for i,word in enumerate(CorpusList):
        #print(i,word)
        # if i = 3; max(3-5,0) and min(3+5,2000)     ----+----+----+
        #           max(0) and min(8)                0 3    8
        # if i = 100; max(100-5,0) and min(100+5,2000)    ----+--100----+----+
        #           max(95) and min(105)                     95    |  105
        # if i = 1998; max(1998-5,0) and min(1998+5,2000)    +----1998-+
        #           max(1993) and min(105)
        for j in range(max(i-window,0),min(i+window,rangeLength)+1): # adding 1, coz loop won't
execute till last interation.
            #print(word,i,CorpusList[j],j)
            #print("Range:",max(i-window,0),min(i+window,rangeLength))
            if (i==j):
                continue #print("---diagonal---")
            else: #if (word==Corpus[j]):
                #print("-------------incrementby1")
                CoMatDF.loc[word,CorpusList[j]]+=1
                #print(CoMatDF)
                #CoMatrix[Corpus[j],word]=CoMatrix[word,Corpus[j]]


window=2
Corpus = "He is not lazy He is intelligent He is smart"
CorpusList=[]
CorpusList=list(Corpus.split(" "))
print(CorpusList)
# ['He', 'is', 'not', 'lazy', 'He', 'is', 'intelligent', 'He', 'is', 'smart']
# --0-----1------2------3------4------5---------6---------7------8------9---

#rangeLength=length+1 #because range func do not include the last interation.
rangeLength=len(CorpusList)-1
print("rangeLength:",rangeLength)
MatColumns=['He', 'is', 'not', 'lazy', 'intelligent', 'smart']

CoMatDF=pd.DataFrame(data=CoMatrix,index=MatColumns,columns=MatColumns)
print(CoMatDF)
#for sentence in tqdm(CorpusList):
    #print("----------------------STARTING---------------------")
    # https://developers.google.com/edu/python/lists
cal_occ(CoMatDF,CorpusList)
print(CoMatDF)
```

```
0.0
(6, 6)
<class 'numpy.ndarray'>
['He', 'is', 'not', 'lazy', 'He', 'is', 'intelligent', 'He', 'is', 'smart']
rangeLength: 9
              He   is  not  lazy  intelligent  smart
He           0.0  0.0  0.0   0.0          0.0    0.0
is           0.0  0.0  0.0   0.0          0.0    0.0
not          0.0  0.0  0.0   0.0          0.0    0.0
lazy         0.0  0.0  0.0   0.0          0.0    0.0
intelligent  0.0  0.0  0.0   0.0          0.0    0.0
smart        0.0  0.0  0.0   0.0          0.0    0.0
              He   is  not  lazy  intelligent  smart
He           0.0  4.0  2.0   1.0          2.0    1.0
is           4.0  0.0  1.0   2.0          2.0    1.0
not          2.0  1.0  0.0   1.0          0.0    0.0
lazy         1.0  2.0  1.0   0.0          0.0    0.0
intelligent  2.0  2.0  0.0   0.0          0.0    0.0
smart        1.0  1.0  0.0   0.0          0.0    0.0
```

In [73]:

```
X_train["EssayTitle"].head(5)
```

Out[73]:

```
7246     my students consist artists ranging pre kinder...
38371    my students amazing full personality each stud...
45304    my fifth graders special they smart little liv...
22689    every day i want students know cared welcome c...
30301    i teach third grade elementary school lexingto...
Name: EssayTitle, dtype: object
```

In [74]:

```
#TFIDF_Feature_EssTitle
#TFIDF_Feature_EssTitle_40
```

In [75]:

```
TFIDF_Feature_EssTitle
```

Out[75]:

	Feature_Word
677	cannot
2392	intelligence
1333	dollars
790	christmas
1225	device
2827	microscope
4616	toward
2185	home
3638	questioners
4798	virginia
4485	teen
3556	production
3991	security
4612	touches
3050	nations
4858	weather
513	benjamin
1227	devoted
334	articles
360	assignment
4975	wrote
717	causing
64	70
390	audio
1549	enough
2164	highlight
4023	sequence
285	anytime

	Feature_Word
4864	weekends
4198	south
...	...
1001	continue
1763	fellow
964	connected
2885	monday
808	classic
714	cause
976	consist
1186	deployed
2132	headsets
1384	drumming
3564	proficient
2225	hugs
787	choosing
500	believing
2171	historic
372	athletes
661	calculators
765	charts
410	aware
1982	genius
1695	external
492	behind
852	codes
1943	funding
1022	cooperatively
2887	monitor
3030	nannansupplies
1597	essentials
1177	demonstrated
608	bring

2000 rows × 1 columns

In [1]:

```
def chk_with_Key_feature_list(text):
    wlist=[]
    #print(text)
    #print(type(text))
    wlist=list(text.split(sep=None))
    # https://stackoverflow.com/questions/14769162/find-matching-words-in-a-list-and-a-string
    if set(wlist).intersection(Key_feature_list):
        return True
    return False
```

In [2]:

```
def cal_occ(CoMatDF,CorpusList,rangeLength,window):
    # https://www.geeksforgeeks.org/enumerate-in-python/
```

```
    # https://www.geeksforgeeks.org/enumerate-in-python/
    for i,word in enumerate(CorpusList):
        #print(i,word)
        #print(type(word))

        if(chk_with_Key_feature_list(word)):
        # if i = 3; max(3-5,0) and min(3+5,2000)     ----+----+----+
        #           max(0) and min(8)                0 3    8
        # if i = 100; max(100-5,0) and min(100+5,2000)    ----+--100----+----+
        #           max(95) and min(105)                     95    |  105
        # if i = 1998; max(1998-5,0) and min(1998+5,2000)    +----1998-+
        #           max(1993) and min(105)
            for j in range(max(i-window,0),min(i+window,rangeLength)+1): # adding 1, coz loop won't
execute till last interation.
                #print(word,i,CorpusList[j],j)
                if(chk_with_Key_feature_list(CorpusList[j])):
                    #print("Range:",max(i-window,0),min(i+window,rangeLength))
                    #if (i!=j):
                    if(word!=CorpusList[j]):
                        CoMatDF.loc[word,CorpusList[j]]+=1
```

In [108]:

```
length=2000
CoMatrix = np.zeros([length,length]) # n is the count of all words
#print(type(TFIDF_Feature_EssTitle))
CoMatDF=pd.DataFrame(data=CoMatrix,index=TFIDF_Feature_EssTitle.Feature_Word,columns=TFIDF_Feature_
EssTitle.Feature_Word)
#print(CoMatDF)

#CoMatDF.loc[word,CorpusList[j]]
#print("1,w",CoMatDF.loc["learners","working"])
#print("1,w",CoMatDF.index("learners",)  loc["learners","working"])


window=5
Key_feature_list=[]
Key_feature_list=list(TFIDF_Feature_EssTitle.Feature_Word)
#print(Key_feature_list)
#print(type(Key_feature_list))

print(CoMatDF)
for sentence in tqdm(X_train["EssayTitle"]):
    CorpusList=[]
    CorpusList=list(sentence.split(" "))
    #print("-"*100)
    #print(CorpusList)
    #print("-"*100)
    rangeLength=len(CorpusList)-1
    #print("rangeLength:",rangeLength)
    cal_occ(CoMatDF,CorpusList,rangeLength,window)
print(CoMatDF)
```

```
Feature_Word    cannot  intelligence  dollars  christmas  device  microscope  \
Feature_Word
cannot             0.0           0.0      0.0        0.0     0.0         0.0
intelligence       0.0           0.0      0.0        0.0     0.0         0.0
dollars            0.0           0.0      0.0        0.0     0.0         0.0
christmas          0.0           0.0      0.0        0.0     0.0         0.0
device             0.0           0.0      0.0        0.0     0.0         0.0
microscope         0.0           0.0      0.0        0.0     0.0         0.0
toward             0.0           0.0      0.0        0.0     0.0         0.0
home               0.0           0.0      0.0        0.0     0.0         0.0
questioners        0.0           0.0      0.0        0.0     0.0         0.0
virginia           0.0           0.0      0.0        0.0     0.0         0.0
teen               0.0           0.0      0.0        0.0     0.0         0.0
production         0.0           0.0      0.0        0.0     0.0         0.0
security           0.0           0.0      0.0        0.0     0.0         0.0
touches            0.0           0.0      0.0        0.0     0.0         0.0
nations            0.0           0.0      0.0        0.0     0.0         0.0
weather            0.0           0.0      0.0        0.0     0.0         0.0
benjamin           0.0           0.0      0.0        0.0     0.0         0.0
devoted            0.0           0.0      0.0        0.0     0.0         0.0
articles           0.0           0.0      0.0        0.0     0.0         0.0
assignment         0.0           0.0      0.0        0.0     0.0         0.0
wrote              0.0           0.0      0.0        0.0     0.0         0.0
```

causing	0.0	0.0	0.0	0.0	0.0	0.0
70	0.0	0.0	0.0	0.0	0.0	0.0
audio	0.0	0.0	0.0	0.0	0.0	0.0
enough	0.0	0.0	0.0	0.0	0.0	0.0
highlight	0.0	0.0	0.0	0.0	0.0	0.0
sequence	0.0	0.0	0.0	0.0	0.0	0.0
anytime	0.0	0.0	0.0	0.0	0.0	0.0
weekends	0.0	0.0	0.0	0.0	0.0	0.0
south	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...
continue	0.0	0.0	0.0	0.0	0.0	0.0
fellow	0.0	0.0	0.0	0.0	0.0	0.0
connected	0.0	0.0	0.0	0.0	0.0	0.0
monday	0.0	0.0	0.0	0.0	0.0	0.0
classic	0.0	0.0	0.0	0.0	0.0	0.0
cause	0.0	0.0	0.0	0.0	0.0	0.0
consist	0.0	0.0	0.0	0.0	0.0	0.0
deployed	0.0	0.0	0.0	0.0	0.0	0.0
headsets	0.0	0.0	0.0	0.0	0.0	0.0
drumming	0.0	0.0	0.0	0.0	0.0	0.0
proficient	0.0	0.0	0.0	0.0	0.0	0.0
hugs	0.0	0.0	0.0	0.0	0.0	0.0
choosing	0.0	0.0	0.0	0.0	0.0	0.0
believing	0.0	0.0	0.0	0.0	0.0	0.0
historic	0.0	0.0	0.0	0.0	0.0	0.0
athletes	0.0	0.0	0.0	0.0	0.0	0.0
calculators	0.0	0.0	0.0	0.0	0.0	0.0
charts	0.0	0.0	0.0	0.0	0.0	0.0
aware	0.0	0.0	0.0	0.0	0.0	0.0
genius	0.0	0.0	0.0	0.0	0.0	0.0
external	0.0	0.0	0.0	0.0	0.0	0.0
behind	0.0	0.0	0.0	0.0	0.0	0.0
codes	0.0	0.0	0.0	0.0	0.0	0.0
funding	0.0	0.0	0.0	0.0	0.0	0.0
cooperatively	0.0	0.0	0.0	0.0	0.0	0.0
monitor	0.0	0.0	0.0	0.0	0.0	0.0
nannansupplies	0.0	0.0	0.0	0.0	0.0	0.0
essentials	0.0	0.0	0.0	0.0	0.0	0.0
demonstrated	0.0	0.0	0.0	0.0	0.0	0.0
bring	0.0	0.0	0.0	0.0	0.0	0.0

Feature_Word	toward	home	questioners	virginia	...	external	behind	\
Feature_Word					...			
cannot	0.0	0.0	0.0	0.0	...	0.0	0.0	
intelligence	0.0	0.0	0.0	0.0	...	0.0	0.0	
dollars	0.0	0.0	0.0	0.0	...	0.0	0.0	
christmas	0.0	0.0	0.0	0.0	...	0.0	0.0	
device	0.0	0.0	0.0	0.0	...	0.0	0.0	
microscope	0.0	0.0	0.0	0.0	...	0.0	0.0	
toward	0.0	0.0	0.0	0.0	...	0.0	0.0	
home	0.0	0.0	0.0	0.0	...	0.0	0.0	
questioners	0.0	0.0	0.0	0.0	...	0.0	0.0	
virginia	0.0	0.0	0.0	0.0	...	0.0	0.0	
teen	0.0	0.0	0.0	0.0	...	0.0	0.0	
production	0.0	0.0	0.0	0.0	...	0.0	0.0	
security	0.0	0.0	0.0	0.0	...	0.0	0.0	
touches	0.0	0.0	0.0	0.0	...	0.0	0.0	
nations	0.0	0.0	0.0	0.0	...	0.0	0.0	
weather	0.0	0.0	0.0	0.0	...	0.0	0.0	
benjamin	0.0	0.0	0.0	0.0	...	0.0	0.0	
devoted	0.0	0.0	0.0	0.0	...	0.0	0.0	
articles	0.0	0.0	0.0	0.0	...	0.0	0.0	
assignment	0.0	0.0	0.0	0.0	...	0.0	0.0	
wrote	0.0	0.0	0.0	0.0	...	0.0	0.0	
causing	0.0	0.0	0.0	0.0	...	0.0	0.0	
70	0.0	0.0	0.0	0.0	...	0.0	0.0	
audio	0.0	0.0	0.0	0.0	...	0.0	0.0	
enough	0.0	0.0	0.0	0.0	...	0.0	0.0	
highlight	0.0	0.0	0.0	0.0	...	0.0	0.0	
sequence	0.0	0.0	0.0	0.0	...	0.0	0.0	
anytime	0.0	0.0	0.0	0.0	...	0.0	0.0	
weekends	0.0	0.0	0.0	0.0	...	0.0	0.0	
south	0.0	0.0	0.0	0.0	...	0.0	0.0	
...	...	...	...	...	...	...	...	
continue	0.0	0.0	0.0	0.0	...	0.0	0.0	
fellow	0.0	0.0	0.0	0.0	...	0.0	0.0	
connected	0.0	0.0	0.0	0.0	...	0.0	0.0	

monday	0.0	0.0	0.0	0.0	...	0.0	0.0
classic	0.0	0.0	0.0	0.0	...	0.0	0.0
cause	0.0	0.0	0.0	0.0	...	0.0	0.0
consist	0.0	0.0	0.0	0.0	...	0.0	0.0
deployed	0.0	0.0	0.0	0.0	...	0.0	0.0
headsets	0.0	0.0	0.0	0.0	...	0.0	0.0
drumming	0.0	0.0	0.0	0.0	...	0.0	0.0
proficient	0.0	0.0	0.0	0.0	...	0.0	0.0
hugs	0.0	0.0	0.0	0.0	...	0.0	0.0
choosing	0.0	0.0	0.0	0.0	...	0.0	0.0
believing	0.0	0.0	0.0	0.0	...	0.0	0.0
historic	0.0	0.0	0.0	0.0	...	0.0	0.0
athletes	0.0	0.0	0.0	0.0	...	0.0	0.0
calculators	0.0	0.0	0.0	0.0	...	0.0	0.0
charts	0.0	0.0	0.0	0.0	...	0.0	0.0
aware	0.0	0.0	0.0	0.0	...	0.0	0.0
genius	0.0	0.0	0.0	0.0	...	0.0	0.0
external	0.0	0.0	0.0	0.0	...	0.0	0.0
behind	0.0	0.0	0.0	0.0	...	0.0	0.0
codes	0.0	0.0	0.0	0.0	...	0.0	0.0
funding	0.0	0.0	0.0	0.0	...	0.0	0.0
cooperatively	0.0	0.0	0.0	0.0	...	0.0	0.0
monitor	0.0	0.0	0.0	0.0	...	0.0	0.0
nannansupplies	0.0	0.0	0.0	0.0	...	0.0	0.0
essentials	0.0	0.0	0.0	0.0	...	0.0	0.0
demonstrated	0.0	0.0	0.0	0.0	...	0.0	0.0
bring	0.0	0.0	0.0	0.0	...	0.0	0.0

Feature_Word	codes	funding	cooperatively	monitor	nannansupplies	\
Feature_Word						
cannot	0.0	0.0	0.0	0.0	0.0	
intelligence	0.0	0.0	0.0	0.0	0.0	
dollars	0.0	0.0	0.0	0.0	0.0	
christmas	0.0	0.0	0.0	0.0	0.0	
device	0.0	0.0	0.0	0.0	0.0	
microscope	0.0	0.0	0.0	0.0	0.0	
toward	0.0	0.0	0.0	0.0	0.0	
home	0.0	0.0	0.0	0.0	0.0	
questioners	0.0	0.0	0.0	0.0	0.0	
virginia	0.0	0.0	0.0	0.0	0.0	
teen	0.0	0.0	0.0	0.0	0.0	
production	0.0	0.0	0.0	0.0	0.0	
security	0.0	0.0	0.0	0.0	0.0	
touches	0.0	0.0	0.0	0.0	0.0	
nations	0.0	0.0	0.0	0.0	0.0	
weather	0.0	0.0	0.0	0.0	0.0	
benjamin	0.0	0.0	0.0	0.0	0.0	
devoted	0.0	0.0	0.0	0.0	0.0	
articles	0.0	0.0	0.0	0.0	0.0	
assignment	0.0	0.0	0.0	0.0	0.0	
wrote	0.0	0.0	0.0	0.0	0.0	
causing	0.0	0.0	0.0	0.0	0.0	
70	0.0	0.0	0.0	0.0	0.0	
audio	0.0	0.0	0.0	0.0	0.0	
enough	0.0	0.0	0.0	0.0	0.0	
highlight	0.0	0.0	0.0	0.0	0.0	
sequence	0.0	0.0	0.0	0.0	0.0	
anytime	0.0	0.0	0.0	0.0	0.0	
weekends	0.0	0.0	0.0	0.0	0.0	
south	0.0	0.0	0.0	0.0	0.0	
...	...	...	...	...	...	
continue	0.0	0.0	0.0	0.0	0.0	
fellow	0.0	0.0	0.0	0.0	0.0	
connected	0.0	0.0	0.0	0.0	0.0	
monday	0.0	0.0	0.0	0.0	0.0	
classic	0.0	0.0	0.0	0.0	0.0	
cause	0.0	0.0	0.0	0.0	0.0	
consist	0.0	0.0	0.0	0.0	0.0	
deployed	0.0	0.0	0.0	0.0	0.0	
headsets	0.0	0.0	0.0	0.0	0.0	
drumming	0.0	0.0	0.0	0.0	0.0	
proficient	0.0	0.0	0.0	0.0	0.0	
hugs	0.0	0.0	0.0	0.0	0.0	
choosing	0.0	0.0	0.0	0.0	0.0	
believing	0.0	0.0	0.0	0.0	0.0	
historic	0.0	0.0	0.0	0.0	0.0	
athletes	0.0	0.0	0.0	0.0	0.0	

calculators	0.0	0.0	0.0	0.0	0.0
charts	0.0	0.0	0.0	0.0	0.0
aware	0.0	0.0	0.0	0.0	0.0
genius	0.0	0.0	0.0	0.0	0.0
external	0.0	0.0	0.0	0.0	0.0
behind	0.0	0.0	0.0	0.0	0.0
codes	0.0	0.0	0.0	0.0	0.0
funding	0.0	0.0	0.0	0.0	0.0
cooperatively	0.0	0.0	0.0	0.0	0.0
monitor	0.0	0.0	0.0	0.0	0.0
nannansupplies	0.0	0.0	0.0	0.0	0.0
essentials	0.0	0.0	0.0	0.0	0.0
demonstrated	0.0	0.0	0.0	0.0	0.0
bring	0.0	0.0	0.0	0.0	0.0

Feature_Word	essentials	demonstrated	bring
Feature_Word			
cannot	0.0	0.0	0.0
intelligence	0.0	0.0	0.0
dollars	0.0	0.0	0.0
christmas	0.0	0.0	0.0
device	0.0	0.0	0.0
microscope	0.0	0.0	0.0
toward	0.0	0.0	0.0
home	0.0	0.0	0.0
questioners	0.0	0.0	0.0
virginia	0.0	0.0	0.0
teen	0.0	0.0	0.0
production	0.0	0.0	0.0
security	0.0	0.0	0.0
touches	0.0	0.0	0.0
nations	0.0	0.0	0.0
weather	0.0	0.0	0.0
benjamin	0.0	0.0	0.0
devoted	0.0	0.0	0.0
articles	0.0	0.0	0.0
assignment	0.0	0.0	0.0
wrote	0.0	0.0	0.0
causing	0.0	0.0	0.0
70	0.0	0.0	0.0
audio	0.0	0.0	0.0
enough	0.0	0.0	0.0
highlight	0.0	0.0	0.0
sequence	0.0	0.0	0.0
anytime	0.0	0.0	0.0
weekends	0.0	0.0	0.0
south	0.0	0.0	0.0
...	...	...	...
continue	0.0	0.0	0.0
fellow	0.0	0.0	0.0
connected	0.0	0.0	0.0
monday	0.0	0.0	0.0
classic	0.0	0.0	0.0
cause	0.0	0.0	0.0
consist	0.0	0.0	0.0
deployed	0.0	0.0	0.0
headsets	0.0	0.0	0.0
drumming	0.0	0.0	0.0
proficient	0.0	0.0	0.0
hugs	0.0	0.0	0.0
choosing	0.0	0.0	0.0
believing	0.0	0.0	0.0
historic	0.0	0.0	0.0
athletes	0.0	0.0	0.0
calculators	0.0	0.0	0.0
charts	0.0	0.0	0.0
aware	0.0	0.0	0.0
genius	0.0	0.0	0.0
external	0.0	0.0	0.0
behind	0.0	0.0	0.0
codes	0.0	0.0	0.0
funding	0.0	0.0	0.0
cooperatively	0.0	0.0	0.0
monitor	0.0	0.0	0.0
nannansupplies	0.0	0.0	0.0
essentials	0.0	0.0	0.0
demonstrated	0.0	0.0	0.0

```
bring                 0.0           0.0    0.0

[2000 rows x 2000 columns]

100%|██████████████████████████████████████████████████████████████████████| 33500/33500
[1:03:09<00:00,  8.76it/s]
```

Feature_Word	cannot	intelligence	dollars	christmas	device	microscope	\
Feature_Word							
cannot	0.0	1.0	0.0	0.0	2.0	0.0	
intelligence	1.0	0.0	0.0	0.0	0.0	0.0	
dollars	0.0	0.0	0.0	0.0	0.0	0.0	
christmas	0.0	0.0	0.0	0.0	0.0	0.0	
device	2.0	0.0	0.0	0.0	0.0	0.0	
microscope	0.0	0.0	0.0	0.0	0.0	0.0	
toward	0.0	2.0	0.0	0.0	4.0	0.0	
home	94.0	2.0	0.0	1.0	20.0	0.0	
questioners	0.0	0.0	0.0	0.0	0.0	0.0	
virginia	0.0	0.0	0.0	0.0	0.0	0.0	
teen	1.0	0.0	0.0	0.0	0.0	0.0	
production	0.0	0.0	0.0	0.0	0.0	0.0	
security	0.0	0.0	0.0	0.0	0.0	0.0	
touches	0.0	0.0	0.0	0.0	1.0	0.0	
nations	0.0	0.0	0.0	0.0	0.0	0.0	
weather	5.0	0.0	0.0	0.0	0.0	0.0	
benjamin	0.0	0.0	0.0	0.0	0.0	0.0	
devoted	0.0	0.0	0.0	0.0	0.0	0.0	
articles	0.0	0.0	0.0	0.0	0.0	0.0	
assignment	1.0	0.0	0.0	1.0	3.0	0.0	
wrote	0.0	0.0	0.0	0.0	0.0	0.0	
causing	0.0	0.0	0.0	0.0	0.0	0.0	
70	0.0	0.0	0.0	0.0	0.0	0.0	
audio	2.0	0.0	0.0	0.0	2.0	0.0	
enough	45.0	0.0	3.0	2.0	10.0	0.0	
highlight	1.0	0.0	0.0	0.0	0.0	0.0	
sequence	0.0	0.0	0.0	0.0	0.0	0.0	
anytime	0.0	0.0	0.0	1.0	0.0	0.0	
weekends	0.0	0.0	0.0	0.0	0.0	0.0	
south	0.0	0.0	0.0	0.0	0.0	0.0	
...	...	...	...	...	...	...	
continue	10.0	2.0	0.0	0.0	1.0	1.0	
fellow	1.0	0.0	0.0	0.0	0.0	0.0	
connected	0.0	0.0	0.0	0.0	2.0	1.0	
monday	0.0	0.0	0.0	0.0	0.0	0.0	
classic	0.0	0.0	0.0	0.0	0.0	0.0	
cause	3.0	0.0	0.0	0.0	1.0	0.0	
consist	0.0	0.0	0.0	0.0	0.0	0.0	
deployed	0.0	0.0	0.0	0.0	0.0	0.0	
headsets	0.0	0.0	0.0	0.0	0.0	0.0	
drumming	0.0	0.0	0.0	0.0	0.0	0.0	
proficient	0.0	0.0	0.0	0.0	0.0	0.0	
hugs	0.0	0.0	0.0	0.0	0.0	0.0	
choosing	1.0	0.0	0.0	0.0	0.0	0.0	
believing	0.0	0.0	0.0	0.0	0.0	1.0	
historic	0.0	0.0	0.0	0.0	0.0	0.0	
athletes	2.0	0.0	0.0	0.0	0.0	0.0	
calculators	7.0	0.0	2.0	0.0	0.0	0.0	
charts	1.0	0.0	0.0	0.0	0.0	0.0	
aware	0.0	1.0	0.0	0.0	0.0	0.0	
genius	0.0	1.0	0.0	0.0	0.0	0.0	
external	2.0	0.0	0.0	0.0	2.0	0.0	
behind	6.0	0.0	0.0	0.0	0.0	0.0	
codes	0.0	0.0	0.0	0.0	1.0	0.0	
funding	16.0	0.0	0.0	0.0	0.0	0.0	
cooperatively	0.0	0.0	0.0	0.0	1.0	0.0	
monitor	0.0	0.0	0.0	0.0	0.0	1.0	
nannansupplies	0.0	0.0	0.0	1.0	0.0	0.0	
essentials	8.0	0.0	0.0	0.0	0.0	0.0	
demonstrated	0.0	0.0	0.0	0.0	0.0	0.0	
bring	39.0	1.0	0.0	0.0	35.0	3.0	

Feature_Word	toward	home	questioners	virginia	...	external	behind	\
Feature_Word					...			
cannot	0.0	94.0	0.0	0.0	...	2.0	6.0	
intelligence	2.0	2.0	0.0	0.0	...	0.0	0.0	
dollars	0.0	0.0	0.0	0.0		0.0	0.0	

dollars	0.0	0.0	0.0	0.0	...	0.0	0.0
christmas	0.0	1.0	0.0	0.0	...	0.0	0.0
device	4.0	20.0	0.0	0.0	...	2.0	0.0
microscope	0.0	0.0	0.0	0.0	...	0.0	0.0
toward	0.0	2.0	0.0	0.0	...	0.0	1.0
home	2.0	0.0	0.0	2.0	...	0.0	21.0
questioners	0.0	0.0	0.0	0.0	...	0.0	0.0
virginia	0.0	2.0	0.0	0.0	...	0.0	0.0
teen	1.0	0.0	0.0	0.0	...	0.0	0.0
production	0.0	0.0	0.0	0.0	...	0.0	2.0
security	0.0	8.0	0.0	0.0	...	0.0	0.0
touches	0.0	0.0	0.0	0.0	...	0.0	0.0
nations	0.0	1.0	0.0	0.0	...	0.0	0.0
weather	0.0	10.0	0.0	0.0	...	0.0	2.0
benjamin	0.0	0.0	0.0	0.0	...	0.0	0.0
devoted	2.0	0.0	0.0	0.0	...	0.0	0.0
articles	1.0	16.0	0.0	0.0	...	0.0	0.0
assignment	0.0	3.0	0.0	0.0	...	0.0	2.0
wrote	0.0	2.0	0.0	0.0	...	0.0	0.0
causing	0.0	2.0	0.0	0.0	...	0.0	1.0
70	0.0	10.0	0.0	2.0	...	0.0	0.0
audio	0.0	10.0	0.0	0.0	...	0.0	0.0
enough	2.0	71.0	0.0	0.0	...	0.0	3.0
highlight	0.0	2.0	0.0	0.0	...	0.0	0.0
sequence	0.0	0.0	0.0	0.0	...	0.0	0.0
anytime	0.0	2.0	0.0	0.0	...	0.0	0.0
weekends	0.0	77.0	0.0	0.0	...	0.0	0.0
south	0.0	6.0	0.0	0.0	...	0.0	0.0
...	...	...	...	...	...	...	...
continue	8.0	107.0	0.0	0.0	...	0.0	9.0
fellow	1.0	1.0	0.0	0.0	...	0.0	1.0
connected	0.0	4.0	0.0	0.0	...	0.0	0.0
monday	0.0	5.0	0.0	0.0	...	0.0	0.0
classic	0.0	3.0	0.0	0.0	...	0.0	0.0
cause	3.0	4.0	0.0	0.0	...	0.0	2.0
consist	0.0	4.0	0.0	0.0	...	0.0	0.0
deployed	0.0	5.0	0.0	0.0	...	0.0	0.0
headsets	0.0	0.0	0.0	0.0	...	0.0	0.0
drumming	0.0	0.0	0.0	0.0	...	0.0	0.0
proficient	2.0	6.0	0.0	0.0	...	0.0	1.0
hugs	0.0	1.0	0.0	0.0	...	0.0	0.0
choosing	2.0	5.0	0.0	0.0	...	0.0	0.0
believing	0.0	1.0	0.0	0.0	...	0.0	1.0
historic	0.0	0.0	0.0	0.0	...	0.0	0.0
athletes	2.0	2.0	0.0	0.0	...	0.0	0.0
calculators	1.0	1.0	0.0	0.0	...	0.0	1.0
charts	1.0	4.0	0.0	0.0	...	0.0	1.0
aware	1.0	2.0	0.0	0.0	...	1.0	4.0
genius	0.0	0.0	0.0	0.0	...	0.0	1.0
external	0.0	0.0	0.0	0.0	...	0.0	0.0
behind	1.0	21.0	0.0	0.0	...	0.0	0.0
codes	0.0	2.0	0.0	0.0	...	0.0	0.0
funding	1.0	27.0	0.0	0.0	...	0.0	3.0
cooperatively	0.0	1.0	0.0	0.0	...	0.0	0.0
monitor	1.0	4.0	0.0	0.0	...	0.0	0.0
nannansupplies	1.0	4.0	0.0	0.0	...	0.0	0.0
essentials	0.0	16.0	0.0	0.0	...	0.0	0.0
demonstrated	0.0	0.0	0.0	0.0	...	0.0	1.0
bring	1.0	266.0	0.0	0.0	...	0.0	3.0

Feature_Word	codes	funding	cooperatively	monitor	nannansupplies	\
Feature_Word						
cannot	0.0	16.0	0.0	0.0	0.0	
intelligence	0.0	0.0	0.0	0.0	0.0	
dollars	0.0	0.0	0.0	0.0	0.0	
christmas	0.0	0.0	0.0	0.0	1.0	
device	1.0	0.0	1.0	0.0	0.0	
microscope	0.0	0.0	0.0	1.0	0.0	
toward	0.0	1.0	0.0	1.0	1.0	
home	2.0	27.0	1.0	4.0	4.0	
questioners	0.0	0.0	0.0	0.0	0.0	
virginia	0.0	0.0	0.0	0.0	0.0	
teen	0.0	0.0	0.0	0.0	0.0	
production	0.0	0.0	0.0	0.0	0.0	
security	0.0	0.0	0.0	0.0	0.0	
touches	1.0	0.0	0.0	0.0	0.0	
nations	0.0	0.0	0.0	0.0	0.0	
weather	0.0	0.0	0.0	1.0	0.0	

weather	0.0	0.0	0.0	1.0	0.0
benjamin	0.0	0.0	0.0	0.0	0.0
devoted	0.0	0.0	0.0	0.0	0.0
articles	0.0	1.0	0.0	0.0	0.0
assignment	1.0	1.0	0.0	0.0	0.0
wrote	0.0	1.0	0.0	0.0	0.0
causing	0.0	1.0	0.0	0.0	0.0
70	0.0	0.0	0.0	0.0	0.0
audio	0.0	1.0	0.0	0.0	1.0
enough	0.0	32.0	0.0	2.0	1.0
highlight	0.0	0.0	0.0	0.0	0.0
sequence	0.0	0.0	1.0	0.0	0.0
anytime	0.0	0.0	0.0	0.0	0.0
weekends	0.0	0.0	0.0	0.0	0.0
south	0.0	2.0	0.0	0.0	0.0
...	...	...	...	...	...
continue	0.0	18.0	0.0	4.0	2.0
fellow	0.0	0.0	0.0	0.0	0.0
connected	0.0	0.0	0.0	0.0	0.0
monday	0.0	0.0	0.0	0.0	0.0
classic	0.0	0.0	0.0	0.0	0.0
cause	0.0	0.0	1.0	0.0	0.0
consist	0.0	0.0	1.0	0.0	0.0
deployed	0.0	0.0	0.0	0.0	0.0
headsets	0.0	0.0	0.0	0.0	0.0
drumming	0.0	2.0	0.0	0.0	0.0
proficient	0.0	0.0	0.0	0.0	0.0
hugs	0.0	0.0	0.0	0.0	0.0
choosing	0.0	0.0	0.0	0.0	0.0
believing	0.0	0.0	0.0	0.0	0.0
historic	0.0	0.0	0.0	0.0	0.0
athletes	0.0	1.0	0.0	1.0	0.0
calculators	0.0	1.0	0.0	0.0	0.0
charts	0.0	0.0	0.0	1.0	0.0
aware	0.0	0.0	0.0	1.0	0.0
genius	0.0	1.0	0.0	0.0	0.0
external	0.0	0.0	0.0	0.0	0.0
behind	0.0	3.0	0.0	0.0	0.0
codes	0.0	0.0	0.0	0.0	0.0
funding	0.0	0.0	0.0	1.0	1.0
cooperatively	0.0	0.0	0.0	0.0	0.0
monitor	0.0	1.0	0.0	0.0	0.0
nannansupplies	0.0	1.0	0.0	0.0	0.0
essentials	0.0	2.0	0.0	0.0	0.0
demonstrated	0.0	0.0	0.0	0.0	0.0
bring	1.0	13.0	1.0	0.0	0.0

Feature_Word	essentials	demonstrated	bring
Feature_Word			
cannot	8.0	0.0	39.0
intelligence	0.0	0.0	1.0
dollars	0.0	0.0	0.0
christmas	0.0	0.0	0.0
device	0.0	0.0	35.0
microscope	0.0	0.0	3.0
toward	0.0	0.0	1.0
home	16.0	0.0	266.0
questioners	0.0	0.0	0.0
virginia	0.0	0.0	0.0
teen	0.0	0.0	0.0
production	0.0	0.0	0.0
security	0.0	0.0	1.0
touches	0.0	0.0	0.0
nations	0.0	0.0	0.0
weather	3.0	0.0	1.0
benjamin	0.0	0.0	0.0
devoted	0.0	0.0	0.0
articles	0.0	0.0	2.0
assignment	0.0	0.0	2.0
wrote	0.0	0.0	0.0
causing	0.0	0.0	0.0
70	0.0	0.0	2.0
audio	0.0	0.0	5.0
enough	1.0	0.0	15.0
highlight	0.0	0.0	4.0
sequence	0.0	0.0	0.0
anytime	0.0	0.0	5.0
weekends	0.0	0.0	3.0

```
weekends                0.0           0.0    3.0
south                   0.0           0.0    2.0
...                     ...           ...    ...
continue                1.0           1.0   22.0
fellow                  0.0           0.0    0.0
connected               0.0           0.0    1.0
monday                  0.0           0.0    1.0
classic                 0.0           0.0    4.0
cause                   0.0           0.0    1.0
consist                 0.0           0.0    1.0
deployed                0.0           0.0    0.0
headsets                0.0           0.0    0.0
drumming                0.0           0.0    1.0
proficient              0.0           0.0    0.0
hugs                    0.0           0.0    0.0
choosing                1.0           0.0    3.0
believing               0.0           0.0    1.0
historic                0.0           0.0    0.0
athletes                0.0           0.0    3.0
calculators             0.0           0.0    3.0
charts                  0.0           0.0    2.0
aware                   0.0           0.0    3.0
genius                  0.0           0.0    1.0
external                0.0           0.0    0.0
behind                  0.0           1.0    3.0
codes                   0.0           0.0    1.0
funding                 2.0           0.0   13.0
cooperatively           0.0           0.0    1.0
monitor                 0.0           0.0    0.0
nannansupplies          0.0           0.0    0.0
essentials              0.0           0.0    3.0
demonstrated            0.0           0.0    0.0
bring                   3.0           0.0    0.0

[2000 rows x 2000 columns]
```

In [110]:

```
countdiagonal=0
for i in range(CoMatDF.shape[0]):
    for j in range(100):
        #print(i,j)
        if (i==j):
            #print("hello")
            if (CoMatDF.iloc[i,j]!=0):
                countdiagonal+=1
            #print(CoMatDF.iloc[i,j])
            #print(CoMatDF[i,j])
print(countdiagonal)
```

```
0
```

Suggestion to check co-occurance matrix, on toy example

In [6]:

```
length=3
CoMatrix = np.zeros([length,length]) # n is the count of all words
#print(type(TFIDF_Feature_EssTitle))
#CoMatDF=pd.DataFrame(data=CoMatrix,index=TFIDF_Feature_EssTitle.Feature_Word,columns=TFIDF_Feature_EssTitle.Feature_Word)
#print(CoMatDF)

#CoMatDF.loc[word,CorpusList[j]]
#print("l,w",CoMatDF.loc["learners","working"])
#print("l,w",CoMatDF.index("learners",)  loc["learners","working"])


window=2
#Key_feature_list=[]
#Key_feature_list=list(TFIDF_Feature_EssTitle.Feature_Word)
#print(Key_feature_list)
#print(type(Key_feature_list))
```

```
CorpusList=["abc def ijk pqr",
        "pqr, klm, opq",
        "lmn pqr xyz abc def pqr abc"]
Key_feature_list= ['abc','pqr','def']

CoMatDF=pd.DataFrame(data=CoMatrix,index=Key_feature_list,columns=Key_feature_list)

print(CoMatDF)
for sentence in tqdm(CorpusList):
    CorpusList=[]
    CorpusList=list(sentence.split(" "))
    #print("-"*100)
    #print(CorpusList)
    #print("-"*100)
    rangeLength=len(CorpusList)-1
    #print("rangeLength:",rangeLength)
    cal_occ(CoMatDF,CorpusList,rangeLength,window)
print(CoMatDF)
```

```
     abc  pqr  def
abc  0.0  0.0  0.0
pqr  0.0  0.0  0.0
def  0.0  0.0  0.0
```

```
100%|██████████| 3/3 [00:00<00:00, 429.66it/s]
```

```
     abc  pqr  def
abc  0.0  3.0  3.0
pqr  3.0  0.0  2.0
def  3.0  2.0  0.0
```

2.3 Applying TruncatedSVD and Calculating Vectors for `essay` and `project_title`

In [111]:

```
# please write all the code with proper documentation, and proper titles for each subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your code
# make sure you featurize train and test data separatly

# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

- **step 3 Use TruncatedSVD on calculated co-occurance matrix and reduce its dimensions, choose the number of components (`n_components`) using elbow method**

  - **The shape of the matrix after TruncatedSVD will be 2000*n, i.e. each row represents a vector form of the corresponding word.**
  - **Vectorize the essay text and project titles using these word vectors. (while vectorizing, do ignore all the words which are not in top 2k words)**

In [112]:

```
type(CoMatDF)
CoMatDF.shape
```

Out[112]:

```
(2000, 2000)
```

In [113]:

```
(CoMatDF != 0).sum(1).sum()
```

Out[113]:

825764

In [114]:

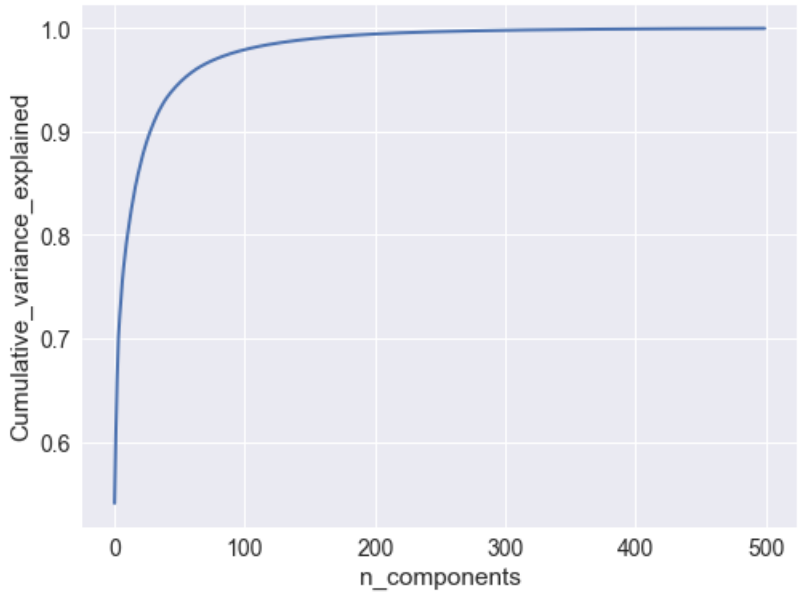
```
np.count_nonzero(CoMatDF)
```

Out[114]:

825764

In [115]:

```
from sklearn.decomposition import TruncatedSVD
svd = TruncatedSVD(n_components = 500)
model = svd.fit_transform(CoMatDF)

percentage_var_explained = svd.explained_variance_ / np.sum(svd.explained_variance_);
cum_var_explained = np.cumsum(percentage_var_explained)

# cumulative explained variance vs n_components
plt.figure(figsize=(8, 6))
plt.plot(cum_var_explained, linewidth=2)
plt.axis()
plt.grid(True)
plt.xlabel('n_components')
plt.ylabel('Cumulative_variance_explained')
plt.show()
```



N=150 covers most of the data variance

In [117]:

```
svd = TruncatedSVD(n_components=150)
result_train=svd.fit_transform(CoMatDF)

print(result_train.shape)
result_train
```

(2000, 150)

Out[117]:

```
array([[ 3.70180304e+02, -4.68649661e+00, -3.52140668e+01, ...,
        -5.67974270e+00, -3.77948981e+00,  9.28241981e+00],
       [ 2.16620311e+01, -3.26874869e+00, -2.17810342e+00, ...,
         9.19176457e-01, -5.30307570e-01, -1.52990700e+00],
       [ 1.48205026e+01,  4.44722422e+00,  1.51532506e+00, ...,
         7.16278445e-01, -3.80623696e-01,  7.26204704e-02],
       ...,
       [ 5.15613884e+01,  6.08035891e-01,  4.93931630e-01, ...,
        -4.20703407e-01, -5.52261448e-01, -9.39379854e-01],
       [ 1.19797022e+01, -6.21354943e-01, -2.96463993e+00, ...,
        -5.85225186e-01,  8.81045222e-02,  1.05807586e-01],
       [ 8.41789816e+02, -5.56456826e+01, -7.06857713e+01, ...,
        -1.54533683e+01,  6.45545227e+00,  3.82922157e+00]])
```

> - **Vectorize the essay text and project titles using these word vectors. (while vectorizing, do ignore all the words which are not in top 2k words)**

In [118]:

```
model = result_train
glove_words = set(df_idf_sort_desc_2k.index)
keys={}
for i,j in enumerate(glove_words):
    #print(i,j)
    keys[j]=i
keys
```

Out[118]:

```
{'firmly': 0,
 'fish': 1,
 'professionals': 2,
 'graduates': 3,
 'raz': 4,
 'reviewing': 5,
 'relaxing': 6,
 'refine': 7,
 'newspapers': 8,
 'presence': 9,
 'contrast': 10,
 'dictionaries': 11,
 'speaks': 12,
 'freshmen': 13,
 'bots': 14,
 '450': 15,
 'comics': 16,
 'carts': 17,
 'constraints': 18,
 'fly': 19,
 'removed': 20,
 'la': 21,
 'nannanon': 22,
 'civil': 23,
 'contest': 24,
 'implementation': 25,
 'inability': 26,
 'collecting': 27,
 'immerse': 28,
 'chickens': 29,
 'chinese': 30,
 'graph': 31,
 'ells': 32,
 'hoops': 33,
 'ipod': 34,
 'dissection': 35,
 'explored': 36,
 'motivational': 37,
 'net': 38,
 'external': 39,
 'assure': 40,
 'honest': 41,
 'volunteers': 42,
```

```
'coloring': 43,
'fitbits': 44,
'gathering': 45,
'ties': 46,
'devoted': 47,
'develops': 48,
'francisco': 49,
'neatly': 50,
'evident': 51,
'chips': 52,
'cups': 53,
'consisting': 54,
'accurate': 55,
'dances': 56,
'cubes': 57,
'birthday': 58,
'soaking': 59,
'courage': 60,
'pulling': 61,
'dangerous': 62,
'prices': 63,
'oftentimes': 64,
'desires': 65,
'stakes': 66,
'pathway': 67,
'lend': 68,
'awards': 69,
'distract': 70,
'conquer': 71,
'lake': 72,
'grip': 73,
'operating': 74,
'toes': 75,
'embark': 76,
'competing': 77,
'hone': 78,
'leg': 79,
'paperless': 80,
'mountain': 81,
'thin': 82,
'surrounds': 83,
'specially': 84,
'transformed': 85,
'thousand': 86,
'retelling': 87,
'electric': 88,
'intellectually': 89,
'aspiring': 90,
'computing': 91,
'boogie': 92,
'tear': 93,
'colleagues': 94,
'assortment': 95,
'combat': 96,
'label': 97,
'elmo': 98,
'renaissance': 99,
'puerto': 100,
'stone': 101,
'closed': 102,
'virginia': 103,
'reaches': 104,
'limiting': 105,
'rights': 106,
'diego': 107,
'figures': 108,
'complain': 109,
'littlebits': 110,
'questioners': 111,
'sedentary': 112,
'fix': 113,
'correlate': 114,
'throwing': 115,
'nets': 116,
'economical': 117,
'heritage': 118,
'requiring': 119,
```

```
'elsewhere': 120,
'norm': 121,
'mature': 122,
'rainbow': 123,
'pants': 124,
'does': 125,
'confined': 126,
'melting': 127,
'counter': 128,
'thousands': 129,
'therapist': 130,
'talked': 131,
'beings': 132,
'transitions': 133,
'nc': 134,
'germs': 135,
'1000': 136,
'250': 137,
'mice': 138,
'presents': 139,
'thrown': 140,
'wisconsin': 141,
'firm': 142,
'embracing': 143,
'files': 144,
'obtaining': 145,
'praise': 146,
'decorate': 147,
'workstations': 148,
'gardening': 149,
'bullying': 150,
'begging': 151,
'ozobot': 152,
'xylophones': 153,
'juniors': 154,
'solar': 155,
'nannancomfy': 156,
'choir': 157,
'female': 158,
'controlled': 159,
'individuality': 160,
'stays': 161,
'museums': 162,
'recordings': 163,
'stages': 164,
'laminated': 165,
'smallest': 166,
'refuse': 167,
'cook': 168,
'changer': 169,
'diary': 170,
'touching': 171,
'teeth': 172,
'bookshelves': 173,
'believer': 174,
'abuse': 175,
'cash': 176,
'de': 177,
'performers': 178,
'unsafe': 179,
'grew': 180,
'file': 181,
'prints': 182,
'cartridges': 183,
'restless': 184,
'kore': 185,
'incorporates': 186,
'bed': 187,
'fascinated': 188,
'citizen': 189,
'graduating': 190,
'historians': 191,
'hooked': 192,
'feature': 193,
'29': 194,
'purposeful': 195,
'rack': 196,
```

```
'resulting': 197,
'94': 198,
'david': 199,
'stickers': 200,
'setup': 201,
'religions': 202,
'yearly': 203,
'representation': 204,
'hardship': 205,
'temperature': 206,
'transforming': 207,
'ict': 208,
'potentially': 209,
'energized': 210,
'montessori': 211,
'trials': 212,
'33': 213,
'playful': 214,
'designers': 215,
'user': 216,
'oil': 217,
'enjoys': 218,
'participated': 219,
'hp': 220,
'compost': 221,
'complicated': 222,
'wow': 223,
'applicable': 224,
'drink': 225,
'worthwhile': 226,
'dvds': 227,
'plain': 228,
'drills': 229,
'election': 230,
'gloves': 231,
'poses': 232,
'tailored': 233,
'repetition': 234,
'unprepared': 235,
'compliment': 236,
'oakland': 237,
'tubs': 238,
'immense': 239,
'exams': 240,
'kentucky': 241,
'representations': 242,
'hi': 243,
'approved': 244,
'rhyming': 245,
'differentiating': 246,
'impairment': 247,
'tall': 248,
'rainy': 249,
'tags': 250,
'witnessed': 251,
'angles': 252,
'fabric': 253,
'schedules': 254,
'john': 255,
'suits': 256,
'newest': 257,
'fans': 258,
'piano': 259,
'era': 260,
'coast': 261,
'disruption': 262,
'franklin': 263,
'happiness': 264,
'nannantime': 265,
'moon': 266,
'war': 267,
'guides': 268,
'celebration': 269,
'pedal': 270,
'treats': 271,
'reminder': 272,
'experimentation': 273,
```

```
'plot': 274,
'stamp': 275,
'costumes': 276,
'nyc': 277,
'colleges': 278,
'bonus': 279,
'advancement': 280,
'films': 281,
'newspaper': 282,
'bursting': 283,
'talkers': 284,
'or': 285,
'parachute': 286,
'follows': 287,
'sending': 288,
'sustain': 289,
'individualize': 290,
'shelving': 291,
'expectation': 292,
'nannanart': 293,
'account': 294,
'nannancoding': 295,
'tickets': 296,
'simulations': 297,
'amounts': 298,
'viewing': 299,
'giant': 300,
'assign': 301,
'shelter': 302,
'boy': 303,
'nannanusing': 304,
'she': 305,
'realistic': 306,
'accurately': 307,
'nannanbook': 308,
'statement': 309,
'multimedia': 310,
'necessarily': 311,
'its': 312,
'nannanchrome': 313,
'virtually': 314,
'organizations': 315,
'emergent': 316,
'92': 317,
'extensive': 318,
'yearn': 319,
'dollars': 320,
'coins': 321,
'ok': 322,
'abc': 323,
'mirror': 324,
'readings': 325,
'arms': 326,
'fallen': 327,
'entertaining': 328,
'cafeteria': 329,
'candy': 330,
'replacing': 331,
'immensely': 332,
'carpets': 333,
'prepares': 334,
'dated': 335,
'filters': 336,
'statuses': 337,
'manners': 338,
'hooks': 339,
'dc': 340,
'matching': 341,
'mindful': 342,
'nutrient': 343,
'bundle': 344,
'concerned': 345,
'arriving': 346,
'nannanfull': 347,
'cubbies': 348,
'ring': 349,
'avenue': 350,
```

```
'nannantech': 351,
'participates': 352,
'button': 353,
'tag': 354,
'wild': 355,
'hygiene': 356,
'entered': 357,
'earning': 358,
'gained': 359,
'citizenship': 360,
'concerns': 361,
'brainstorm': 362,
'established': 363,
'48': 364,
'scenarios': 365,
'gotten': 366,
'celebrating': 367,
'sad': 368,
'fidgets': 369,
'dark': 370,
'detroit': 371,
'excites': 372,
'speaker': 373,
'rings': 374,
'nannanyou': 375,
'shake': 376,
'predominately': 377,
'embraces': 378,
'disruptive': 379,
'maryland': 380,
'nannanactive': 381,
'seventy': 382,
'guarantee': 383,
'attached': 384,
'positivity': 385,
'microphones': 386,
'outgoing': 387,
'worse': 388,
'earlier': 389,
'ancient': 390,
'bathroom': 391,
'toon': 392,
'ambassadors': 393,
'opposed': 394,
'carrying': 395,
'recognizing': 396,
'nannanorganization': 397,
'stepping': 398,
'alternatives': 399,
'wasting': 400,
'cds': 401,
'flooded': 402,
'worrying': 403,
'metal': 404,
'expo': 405,
'grandparent': 406,
'aspirations': 407,
'evaluate': 408,
'encompasses': 409,
'demonstrations': 410,
'subtracting': 411,
'holiday': 412,
'bluetooth': 413,
'linguistically': 414,
'ny': 415,
'robotic': 416,
'monitors': 417,
'pennsylvania': 418,
'grows': 419,
'restricted': 420,
'greek': 421,
'invention': 422,
'host': 423,
'operate': 424,
'tells': 425,
'hula': 426,
'pbl': 427,
```

```
'agreed': 428,
'recycle': 429,
'gel': 430,
'empowers': 431,
'wearing': 432,
'stamps': 433,
'tears': 434,
'agricultural': 435,
'breaking': 436,
'partnership': 437,
'hub': 438,
'softball': 439,
'uniquely': 440,
'oils': 441,
'hydrated': 442,
'adequately': 443,
'beds': 444,
'cardboard': 445,
'surprised': 446,
'dimensional': 447,
'asd': 448,
'buzz': 449,
'weapon': 450,
'staple': 451,
'buttons': 452,
'honors': 453,
'symbols': 454,
'mindfulness': 455,
'prison': 456,
'nurtured': 457,
'pillow': 458,
'such': 459,
'dramatically': 460,
'occupational': 461,
'ear': 462,
'bi': 463,
'empathetic': 464,
'ribbon': 465,
'halls': 466,
'firsthand': 467,
'masterpiece': 468,
'mood': 469,
'wires': 470,
'network': 471,
'cabinet': 472,
'lock': 473,
'entertainment': 474,
'celebrated': 475,
'impacting': 476,
'tile': 477,
'fundraising': 478,
'workspace': 479,
'wood': 480,
'ice': 481,
'outlook': 482,
'treasure': 483,
'links': 484,
'recycling': 485,
'nannanchromebook': 486,
'vehicle': 487,
'tied': 488,
'spirited': 489,
'residents': 490,
'fewer': 491,
'impressed': 492,
'occasion': 493,
'television': 494,
'clues': 495,
'dominican': 496,
'kept': 497,
'inspirations': 498,
'certified': 499,
'newcomers': 500,
'adapted': 501,
'announcements': 502,
'promise': 503,
'window': 504,
```

```
'approaches': 505,
'personalize': 506,
'proactive': 507,
'habitat': 508,
'sun': 509,
'stopped': 510,
'lacrosse': 511,
'dollar': 512,
'explaining': 513,
'enabling': 514,
'eighty': 515,
'burden': 516,
'ordered': 517,
'profession': 518,
'observed': 519,
'examine': 520,
'accounts': 521,
'museum': 522,
'intrigued': 523,
'switch': 524,
'globally': 525,
'blossom': 526,
'avenues': 527,
'strides': 528,
'therapeutic': 529,
'region': 530,
'historic': 531,
'illustrators': 532,
'hair': 533,
'stomach': 534,
'arizona': 535,
'headsets': 536,
'cease': 537,
'remains': 538,
'car': 539,
'demographic': 540,
'girl': 541,
'resiliency': 542,
'sat': 543,
'initial': 544,
'exception': 545,
'dividers': 546,
'beloved': 547,
'ocean': 548,
'unfortunate': 549,
'150': 550,
'mark': 551,
'whisper': 552,
'western': 553,
'buildings': 554,
'rising': 555,
'housed': 556,
'drawer': 557,
'divide': 558,
'unless': 559,
'scarce': 560,
'sample': 561,
'surely': 562,
'cater': 563,
'gender': 564,
'occurs': 565,
'harsh': 566,
'qualities': 567,
'transformation': 568,
'regards': 569,
'guitars': 570,
'relatives': 571,
'protected': 572,
'breathing': 573,
'reviews': 574,
'capability': 575,
'setbacks': 576,
'quest': 577,
'disturbing': 578,
'mandarin': 579,
'lovers': 580,
'kahoot': 581,
```

```
'nannantake': 582,
'shaped': 583,
'disc': 584,
'epic': 585,
'flowers': 586,
'produced': 587,
'former': 588,
'advances': 589,
'stated': 590,
'modified': 591,
'frog': 592,
'bell': 593,
'3doodler': 594,
'nannansensory': 595,
'acting': 596,
'flag': 597,
'hurt': 598,
'phase': 599,
'crisis': 600,
'brainstorming': 601,
'discs': 602,
'blending': 603,
'cube': 604,
'illinois': 605,
'expanded': 606,
'basically': 607,
'nook': 608,
'heroes': 609,
'brighten': 610,
'knees': 611,
'zoo': 612,
'belonging': 613,
'scenes': 614,
'absent': 615,
'peaceful': 616,
'brainstormed': 617,
'king': 618,
'trackers': 619,
'situated': 620,
'backs': 621,
'very': 622,
'arkansas': 623,
'nannan21st': 624,
'visiting': 625,
'grouping': 626,
'soap': 627,
'blog': 628,
'shortage': 629,
'showed': 630,
'themed': 631,
'mississippi': 632,
'eyed': 633,
'drawn': 634,
'passions': 635,
'frequency': 636,
'albert': 637,
'apparent': 638,
'nannanlisten': 639,
'fold': 640,
'wealthy': 641,
'southwest': 642,
'camp': 643,
'windows': 644,
'characteristics': 645,
'pitch': 646,
'tennessee': 647,
'pbis': 648,
'utensils': 649,
'beginner': 650,
'requests': 651,
'likes': 652,
'dictionary': 653,
'vivid': 654,
'reflects': 655,
'courageous': 656,
'bass': 657,
'thoroughly': 658,
```

```
'pouches': 659,
'achievers': 660,
'boston': 661,
'integrity': 662,
'hats': 663,
'latin': 664,
'drums': 665,
'solely': 666,
'scene': 667,
'drawers': 668,
'delight': 669,
'nannanhealthy': 670,
'ebooks': 671,
'writings': 672,
'pattern': 673,
'inventors': 674,
'classmate': 675,
'mild': 676,
'eggs': 677,
'recommended': 678,
'houston': 679,
'nannando': 680,
'rain': 681,
'detailed': 682,
'humans': 683,
'measuring': 684,
'ended': 685,
'chaotic': 686,
'liked': 687,
'steady': 688,
'fluorescent': 689,
'macbook': 690,
'greeted': 691,
'entry': 692,
'tirelessly': 693,
'parks': 694,
'polite': 695,
'vocational': 696,
'el': 697,
'cares': 698,
'historically': 699,
'utmost': 700,
'ensures': 701,
'skin': 702,
'screens': 703,
'invent': 704,
'trays': 705,
'riding': 706,
'worried': 707,
'subtract': 708,
'packed': 709,
'cleaner': 710,
'mechanical': 711,
'nannancolor': 712,
'diploma': 713,
'fundamentals': 714,
'brainpop': 715,
'placing': 716,
'appeal': 717,
'youtube': 718,
'neighbors': 719,
'reminders': 720,
'deployed': 721,
'flowing': 722,
'nannanliteracy': 723,
'safer': 724,
'hall': 725,
'volleyballs': 726,
'toy': 727,
'dallas': 728,
'george': 729,
'phoenix': 730,
'logical': 731,
'conditioning': 732,
'ngss': 733,
'researchers': 734,
'explained': 735
```

```
'explained': 735,
'intend': 736,
'nannanstudent': 737,
'storyworks': 738,
'supplied': 739,
'introduction': 740,
'minecraft': 741,
'delay': 742,
'clock': 743,
'baltimore': 744,
'wondering': 745,
'retell': 746,
'street': 747,
'97': 748,
'sunshine': 749,
'traveling': 750,
'factor': 751,
'object': 752,
'cones': 753,
'scratch': 754,
'treated': 755,
'bulletin': 756,
'orchestra': 757,
'counseling': 758,
'populations': 759,
'nannankeep': 760,
'scholar': 761,
'respected': 762,
'crayon': 763,
'benches': 764,
'holidays': 765,
'traditionally': 766,
'calendar': 767,
'alleviate': 768,
'bouncing': 769,
'coping': 770,
'fairly': 771,
'broadcast': 772,
'versions': 773,
'followed': 774,
'cooperate': 775,
'flex': 776,
'humble': 777,
'predict': 778,
'formed': 779,
'balancing': 780,
'spatial': 781,
'minimize': 782,
'identification': 783,
'survival': 784,
'paths': 785,
'recorded': 786,
'plethora': 787,
'roller': 788,
'settle': 789,
'pta': 790,
'inventions': 791,
'conscious': 792,
'session': 793,
'wise': 794,
'nannansteam': 795,
'collar': 796,
'indoors': 797,
'closet': 798,
'happened': 799,
'bond': 800,
'helpers': 801,
'kinders': 802,
'company': 803,
'political': 804,
'coaches': 805,
'joining': 806,
'fishing': 807,
'locks': 808,
'khan': 809,
'beads': 810,
'tales': 811,
'strings': 812
```

```
'strings': 812,
'twelve': 813,
'batteries': 814,
'generate': 815,
'conducting': 816,
'saved': 817,
'pivotal': 818,
'producing': 819,
'crackers': 820,
'pushes': 821,
'organizer': 822,
'intrinsic': 823,
'graphics': 824,
'protection': 825,
'plate': 826,
'height': 827,
'insight': 828,
'nannanplease': 829,
'detail': 830,
'seed': 831,
'nationalities': 832,
'land': 833,
'dress': 834,
'tolerance': 835,
'laboratory': 836,
'responding': 837,
'nex': 838,
'violin': 839,
'indianapolis': 840,
'chains': 841,
'ecosystems': 842,
'permanent': 843,
'involving': 844,
'frames': 845,
'submit': 846,
'demonstration': 847,
'lift': 848,
'dad': 849,
'recipe': 850,
'propel': 851,
'collage': 852,
'crazy': 853,
'organic': 854,
'fashion': 855,
'timers': 856,
'preserve': 857,
'subscriptions': 858,
'touches': 859,
'lie': 860,
'grants': 861,
'comparing': 862,
'plates': 863,
'hallway': 864,
'privacy': 865,
'imovie': 866,
'evolving': 867,
'traveled': 868,
'fairy': 869,
'link': 870,
'ninth': 871,
'sweetest': 872,
'practiced': 873,
'compose': 874,
'embraced': 875,
'equitable': 876,
'which': 877,
'observing': 878,
'trash': 879,
'baccalaureate': 880,
'finances': 881,
'peace': 882,
'dojo': 883,
'chaos': 884,
'tricky': 885,
'specials': 886,
'nannanall': 887,
'selections': 888,
'senior': 889
```

```
'senior': 889,
'newer': 890,
'invest': 891,
'believing': 892,
'injury': 893,
'atlanta': 894,
'okay': 895,
'greenhouse': 896,
'concerts': 897,
'law': 898,
'satisfy': 899,
'bases': 900,
'realities': 901,
'unfamiliar': 902,
'distance': 903,
'tradition': 904,
'brushes': 905,
'recorders': 906,
'tinker': 907,
'mystery': 908,
'encountered': 909,
'lakeshore': 910,
'ending': 911,
'reeds': 912,
'beanbags': 913,
'nannanwriting': 914,
'resilience': 915,
'articulation': 916,
'glass': 917,
'succeeding': 918,
'scary': 919,
'hinder': 920,
'dots': 921,
'consistency': 922,
'benjamin': 923,
'representing': 924,
'visible': 925,
'drug': 926,
'adulthood': 927,
'counselor': 928,
'contact': 929,
'acquired': 930,
'guiding': 931,
'largely': 932,
'ozobots': 933,
'alike': 934,
'teen': 935,
'receptive': 936,
'nannanproject': 937,
'enormous': 938,
'planting': 939,
'falls': 940,
'establishing': 941,
'cerebral': 942,
'dough': 943,
'profound': 944,
'hawaiian': 945,
'somewhat': 946,
'magnificent': 947,
'embedded': 948,
'bank': 949,
'gangs': 950,
'versus': 951,
'bears': 952,
'coffee': 953,
'dynamics': 954,
'overlooked': 955,
'fell': 956,
'imagined': 957,
'fifteen': 958,
'bottle': 959,
'differing': 960,
'disciplines': 961,
'worries': 962,
'cricut': 963,
'spectacular': 964,
'thru': 965,
'buddies': 966
```

```
'puggies': 966,
'hockey': 967,
'battery': 968,
'explorations': 969,
'sleep': 970,
'flight': 971,
'headphone': 972,
'nearby': 973,
'adorable': 974,
'apartment': 975,
'genius': 976,
'degrees': 977,
'anatomy': 978,
'string': 979,
'natives': 980,
'bells': 981,
'protectors': 982,
'puppets': 983,
'bike': 984,
'mexican': 985,
'lecture': 986,
'stretching': 987,
'arrived': 988,
'unstable': 989,
'rent': 990,
'continuous': 991,
'walked': 992,
'dice': 993,
'farms': 994,
'meditation': 995,
'beanbag': 996,
'easels': 997,
'listeners': 998,
'replenish': 999,
...}
```

Make Data Model Ready: project_essay | AVG W2V

In [119]:

```
# average Word2Vec for Train Essay
# compute average word2vec for each review.
X_train_essay_avg_w2v = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_train['essay'].values): # for each review/sentence
    vector = np.zeros(150) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words: #glove_word is a set
            vector += model[keys[word]]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    X_train_essay_avg_w2v.append(vector)

print(len(X_train_essay_avg_w2v))
print(len(X_train_essay_avg_w2v[0]))
```

```
100%|████████████████████████████████████████████████████████████████| 33500/33500
[00:02<00:00, 15177.99it/s]
```

```
33500
150
```

In [120]:

```
# average Word2Vec for Test Essay
# compute average word2vec for each review.
X_test_essay_avg_w2v = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_test['essay'].values): # for each review/sentence
    vector = np.zeros(150) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
```

```
            vector += model[keys[word]]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    X_test_essay_avg_w2v.append(vector)

print(len(X_test_essay_avg_w2v))
print(len(X_test_essay_avg_w2v[0]))
```

```
100%|████████████████████████████████████████████████████████████| 16500/16500
[00:00<00:00, 17126.24it/s]
```

```
16500
150
```

Make Data Model Ready: project_title | AVG W2V

In [121]:

```
# average Word2Vec for Train Title
# compute average word2vec for each review.
X_train_title_avg_w2v = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_train['project_title'].values): # for each review/sentence
    vector = np.zeros(150) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[keys[word]]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    X_train_title_avg_w2v.append(vector)

print(len(X_train_title_avg_w2v))
print(len(X_train_title_avg_w2v[0]))
```

```
100%|████████████████████████████████████████████████████████████| 33500/33500
[00:00<00:00, 191402.03it/s]
```

```
33500
150
```

In [122]:

```
# average Word2Vec for Test Essay
# compute average word2vec for each review.
X_test_title_avg_w2v = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_test['project_title'].values): # for each review/sentence
    vector = np.zeros(150) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[keys[word]]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    X_test_title_avg_w2v.append(vector)

print(len(X_test_title_avg_w2v))
print(len(X_test_title_avg_w2v[0]))
```

```
100%|████████████████████████████████████████████████████████████| 16500/16500
[00:00<00:00, 226678.29it/s]
```

```
16500
150
```

2.4 Merge the features from step 3 and step 4

2.4 Merge the features from Step 3 and Step 4

In [123]:

```
# please write all the code with proper documentation, and proper titles for each subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your code
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

In [124]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_tr_set5_GBT = hstack((X_train_state_ohe, X_train_clean_ohe, X_train_cleanSub_ohe,
X_train_grade_ohe,X_train_teacher_ohe, X_train_quantity_norm, X_train_TprevPrj_norm,
X_train_price_norm,X_train_neg_norm,X_train_pos_norm,X_train_neu_norm,X_train_compound_norm,
X_train_title_wc_norm,X_train_essay_wc_norm,X_train_essay_avg_w2v,X_train_title_avg_w2v)).tocsr()
X_te_set5_GBT = hstack((X_test_state_ohe, X_test_clean_ohe,X_test_cleanSub_ohe, X_test_grade_ohe,X_
test_teacher_ohe, X_test_quantity_norm, X_test_TprevPrj_norm, X_test_price_norm, X_test_neg_norm,X
_test_pos_norm,X_test_neu_norm,X_test_compound_norm,X_test_title_wc_norm, X_test_essay_wc_norm,X_t
est_essay_avg_w2v,X_test_title_avg_w2v)).tocsr()

print("Final Data matrix | XGBOOST")
print(X_tr_set5_GBT.shape, y_train.shape)
print(X_te_set5_GBT.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix | XGBOOST
(33500, 408) (33500,)
(16500, 408) (16500,)
====================================================================================================
```

2.5 Apply XGBoost on the Final Features from the above section

https://xgboost.readthedocs.io/en/latest/python/python_intro.html

In [125]:

```
# No need to split the data into train and test(cv)
# use the Dmatrix and apply xgboost on the whole data
# please check the Quora case study notebook as reference

# please write all the code with proper documentation, and proper titles for each subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your code
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

In [126]:

```
import sys
import math

import numpy as np
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import roc_auc_score

# you might need to install this one
import xgboost as xgb

class XGBoostClassifier():
    def __init__(self, num_boost_round=10, **params):
```

```
    def __init__(self, num_boost_round=10, **params):
        self.clf = None
        self.num_boost_round = num_boost_round
        self.params = params
        self.params.update({'objective': 'multi:softprob'})

    def fit(self, X, y, num_boost_round=None):
        num_boost_round = num_boost_round or self.num_boost_round
        self.label2num = {label: i for i, label in enumerate(sorted(set(y)))}
        dtrain = xgb.DMatrix(X, label=[self.label2num[label] for label in y])
        self.clf = xgb.train(params=self.params, dtrain=dtrain, num_boost_round=num_boost_round, ve
rbose_eval=1)

    def predict(self, X):
        num2label = {i: label for label, i in self.label2num.items()}
        Y = self.predict_proba(X)
        y = np.argmax(Y, axis=1)
        return np.array([num2label[i] for i in y])

    def predict_proba(self, X):
        dtest = xgb.DMatrix(X)
        return self.clf.predict(dtest)

    def score(self, X, y):
        Y = self.predict_proba(X)[:,1]
        return roc_auc_score(y, Y)

    def get_params(self, deep=True):
        return self.params

    def set_params(self, **params):
        if 'num_boost_round' in params:
            self.num_boost_round = params.pop('num_boost_round')
        if 'objective' in params:
            del params['objective']
        self.params.update(params)
        return self
```

In [127]:

```
XGclf = XGBoostClassifier(eval_metric = 'auc', num_class = 2, nthread = 4)
########################################################################
#               Change from here                                       #
########################################################################
parameters = {
    'num_boost_round': [5,11,15,21,25], #[100, 250, 500],
    'eta': [0.05, 0.1, 0.3],
    'max_depth': [2,3,5,7,10],  #[6, 9, 12],
    'subsample': [0.9, 1.0],
    'colsample_bytree': [0.9, 1.0],
}

clf = GridSearchCV(XGclf, parameters,cv=3, scoring='roc_auc', return_train_score=True)
# return_train_score : boolean, default=False
# If False, the cv_results_ attribute will not include training scores. Computing training scores
is used to
# get insights on how different parameter settings impact the overfitting/underfitting trade-off.
However computing
# the scores on the training set can be computationally expensive and is not strictly required to
select the parameters
# that yield the best generalization performance.


#X = np.array([[1,2], [3,4], [2,1], [4,3], [1,0], [4,5]])
#Y = np.array([0, 1, 0, 1, 0, 1])
clf.fit(X_tr_set5_GBT, y_train)
```

Out[127]:

```
GridSearchCV(cv=3, error_score='raise-deprecating',
             estimator=<__main__.XGBoostClassifier object at 0x0000018AA2D38BE0>,
             iid='warn', n_jobs=None,
             param_grid={'colsample_bytree': [0.9, 1.0],
                         'eta': [0.05, 0.1, 0.3], 'max_depth': [2, 3, 5, 7, 10],
                         'num_boost_round': [5, 11, 15, 21, 25],
                         'subsample': [0.9, 1.0]},
```

```
            pre_dispatch='2*n_jobs', refit=True, return_train_score=True,
            scoring='roc_auc', verbose=0)
```

In [128]:

```
print(clf.best_estimator_)
print(clf.best_params_)
print(clf.score(X_te_set5_GBT, y_test))
```

```
<__main__.XGBoostClassifier object at 0x0000018AA86E87B8>
{'colsample_bytree': 1.0, 'eta': 0.3, 'max_depth': 3, 'num_boost_round': 25, 'subsample': 0.9}
0.5731184586861037
```

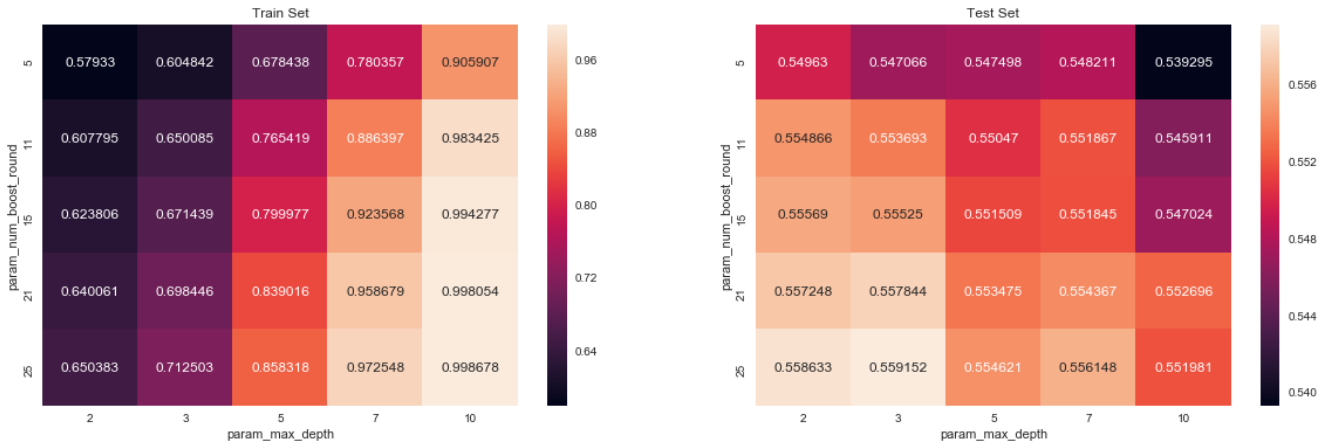
In [129]:

```
#clf.cv_results_
#clf.
print(clf.best_estimator_)
```

```
<__main__.XGBoostClassifier object at 0x0000018AA86E87B8>
```

In [130]:

```
# from Assignment 8_DonorsChoose_DT

# https://seaborn.pydata.org/generated/seaborn.heatmap.html
import seaborn as sns; sns.set()
max_scores1=pd.DataFrame(clf.cv_results_).groupby(['param_num_boost_round','param_max_depth']).max
().unstack()[['mean_test_score','mean_train_score']]

fig,ax=plt.subplots(1,2,figsize=(20,6))

sns.heatmap(max_scores1.mean_train_score,annot=True,fmt='4g',ax=ax[0])
sns.heatmap(max_scores1.mean_test_score,annot=True,fmt='4g',ax=ax[1])

ax[0].set_title('Train Set')
ax[1].set_title('Test Set')

plt.show()
```



In [131]:

```
print(clf.score(X_tr_set5_GBT,y_train))
print(clf.score(X_te_set5_GBT,y_test))
print(clf.best_params_)
print(clf.best_score_)
```

```
0.6802686578348542
0.5731184586861037
{'colsample_bytree': 1.0, 'eta': 0.3, 'max_depth': 3, 'num_boost_round': 25, 'subsample': 0.9}
0.5591520226705807
```

Best Parameter

'max_depth': 2, 'num_boost_round': 11

In [132]:

```
#Best tune parameters
param_grid = {'max_depth': [3],
              'num_boost_round': [25]
             }
```

In [133]:

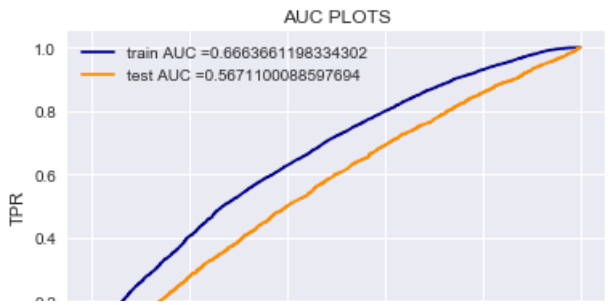
```
#code source:
http://occam.olin.edu/sites/default/files/DataScienceMaterials/machine_learning_lecture_2/Machine%2
rning%20Lecture%202.html
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.datasets import *
import xgboost as xgb

#Using GridSearchCV
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
# XGBoostError: value 0 for Parameter num_class should be greater equal to 1
modelbestXBb = GridSearchCV(XGBoostClassifier(num_class = 2),param_grid)
modelbestXBb.fit(X_tr_set5_GBT, y_train)

print(modelbestXBb.best_score_)
print(modelbestXBb.score(X_te_set5_GBT, y_test))
```

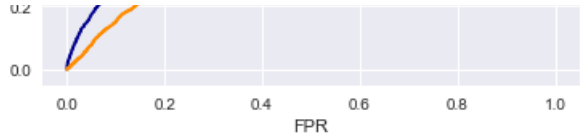
```
0.5524548681147534
0.5671100088597694
```

In [134]:

```
# https://scikit-
learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc

y_train_XB_pred = modelbestXBb.predict_proba(X_tr_set5_GBT)[:,1]
y_test_XB_pred = modelbestXBb.predict_proba(X_te_set5_GBT)[:,1]

print(modelbestXBb.best_params_)
print(modelbestXBb.score(X_te_set5_GBT, y_test))

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_XB_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_XB_pred)

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)),color='darkblue'
)
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)),color='darkorange')
plt.legend()
plt.xlabel("FPR")
plt.ylabel("TPR")
plt.title("AUC PLOTS")
plt.grid(True)
plt.show()
```

```
{'max_depth': 3, 'num_boost_round': 25}
0.5671100088597694
```





In [135]:

```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def predict(proba, threshould, fpr, tpr):

    t = threshould[np.argmax(tpr*(1-fpr))]

    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is very high

    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshold", np.round(t,3))
    predictions = []
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

In [136]:

```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_XB_pred, tr_thresholds, train_fpr, train_tpr)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_XB_pred, te_thresholds, test_fpr, test_tpr)))
```

```
====================================================================================================

Train confusion matrix
the maximum value of tpr*(1-fpr) 0.37932621888453105 for threshold 0.846
[[ 3340  1828]
 [11703 16629]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.30358997991492864 for threshold 0.847
[[1378 1168]
 [6127 7827]]
```

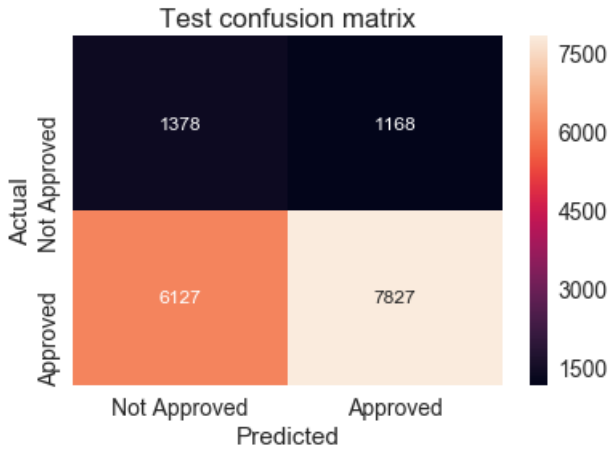
In [137]:

```
import seaborn as snTr
import seaborn as snTe
import pandas as pdH
import matplotlib.pyplot as pltTr
import matplotlib.pyplot as pltTe

# https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix
arrayTr=confusion_matrix(y_train, predict(y_train_XB_pred, tr_thresholds, train_fpr, train_tpr))
df_cmTr = pdH.DataFrame(arrayTr,range(2),range(2))
#print(arrayTr)
# https://stackoverflow.com/questions/32723798/how-do-i-add-a-title-to-seaborn-heatmap
axTr = pltTr.axes()

snTr.set(font_scale=1.4)#for label size

# https://seaborn.pydata.org/generated/seaborn.heatmap.html

snTr.heatmap(df_cmTr, annot=True,annot_kws={"size": 12},fmt="d",ax=axTr)# font size, format in
digit

labels=['Not Approved','Approved']
axTr.set_xticklabels(labels)
axTr.set_yticklabels(labels)
#Suggestion 4.Label confusion matrix heatmap with actual and predicted labels.
pltTr.title("Train confusion matrix")
```

```
pltTr.xlabel("Predicted")
pltTr.ylabel("Actual")
pltTr.show()

# https://stackoverflow.com/questions/50947776/plot-two-seaborn-heatmap-graphs-side-by-side
#fig, ax =plt.subplots(1,1)

# https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix
arrayTe=confusion_matrix(y_test, predict(y_test_XB_pred, te_thresholds, test_fpr, test_tpr))
df_cmTe = pdH.DataFrame(arrayTe,range(2),range(2))

axTe = pltTe.axes()

snTe.set(font_scale=1.4)#for label size

# https://seaborn.pydata.org/generated/seaborn.heatmap.html
snTe.heatmap(df_cmTe, annot=True,annot_kws={"size": 12},fmt="d",ax=axTe)# font size, format in
digit

#Suggestion 4.Label confusion matrix heatmap with actual and predicted labels.
axTe.set_xticklabels(labels)
axTe.set_yticklabels(labels)
pltTe.title("Test confusion matrix")
pltTe.xlabel("Predicted")
pltTe.ylabel("Actual")
pltTe.show()
```

```
the maximum value of tpr*(1-fpr) 0.37932621888453105 for threshold 0.846
```


```
the maximum value of tpr*(1-fpr) 0.30358997991492864 for threshold 0.847
```



3. Conclusion

In [138]:

```
# Please write down few lines about what you observed from this assignment.
# Please compare all your models using Prettytable library
```

```
from prettytable import PrettyTable

x = PrettyTable()

x.field_names = ["Vectorizer", "Model", "max_depth", "num_boost_round","AUC"]

x.add_row(["wordtovec", "XgBoost  ", 2, 11, 0.5671100088597694 ])

print(x)
```

```
+------------+-----------+-----------+-----------------+--------------------+
| Vectorizer |   Model   | max_depth | num_boost_round |        AUC         |
+------------+-----------+-----------+-----------------+--------------------+
| wordtovec  | XgBoost   |     2     |        11       | 0.5671100088597694 |
+------------+-----------+-----------+-----------------+--------------------+
```

Summary

Step followed

- **Preprocessing of Project_subject_categories**
- **Preprocessing of Project_subject_subcategories**
- **Preprocessing of Project_grade_category**
- **Preprocessing of teacher_prefix**
- **Text Preprocessing for Project essay and Project Title**
- **Numeric feature for Essay, no of wordcount**
- **Numeric feature for Project Title, no of wordcount**
- **Compute Sentiment score**
  - **'neg', 'pos', 'neu', 'compound'**
- **Add Numeric features (preprocessed_essay, preprocessed_title), Essay word count, Project's title word count, Compute Sentiment score('neg', 'pos', 'neu', 'compound') in project_data**
- **Took first 50000 data points for doing the assignment # and removed the Class lable (Project_is_approved)**
- **Split the data in Train and Test**

Making datamodel ready

text

```
        - encoding of school_state is splited into Train and Test vector
        - encoding of clean_category is splited into Train and Test vector
        - encoding of clean_subcategory is splited into Train and Test vector
        - encoding of project_grade_category is splited into Train and Test vector
        - encoding of teacher_prefix is splited into Train and Test vector
```

numeric

```
        - encoding of quantity is splited into Train and Test vector
        - encoding of teacher_number_of_previously_posted_projects is splited into Train and
  Test vector
        - encoding of price is splited into Train and Test vector
        - encoding of sentimental score | neg, is splited into Train and Test vector
        - encoding of sentimental score | pos, is splited into Train and Test vector
        - encoding of sentimental score | neu, is splited into Train and Test vector
        - encoding of sentimental score | compound, is splited into Train and Test vector
        - encoding of numerical | number of words in the title, is splited into Train and
  Test vector
        - encoding of numerical | number of words in the essay, is splited into Train and
  Test vector
```

- **concatanate essay text with project title in EssayTitle and then find the top 2k words**
  - **Vectorize EssayTitle, with TFIDFVectorizer.**
  - **Made a dataframe with, Tfidf_vectorizer.get_feature_names() as index and Tfidf *vectorizer.idf* as columns**
  - **sort with argsort() to sort with index**

- Take top 2000 words

-Create Co-Occurance MAtrix

```
- Function chk_with_Key_feature_list: This take Text as input, and return true, if it exist
in Key_feature_list
- Function cal_occ: this take whole Dataframe, Corpus list, rangeLength, Window=5
    For each word in CorpusList,
        it check if the word is present in Key_feature_list, by calling
chk_with_Key_feature_list function
            if the word exist: For a window of 5 word, both left and right side of the data,
it check
                it call chk_with_Key_feature_list function, and check its neigbour (here 5)
one by one
                    if i=j, i,e, it is a diagonal matrix, then ignore
                    else : increment the count of the Dataframe for that i,j cell
- numpy matrix of all zero is initialize
- Datafrmae is created on above numpy matrix, with Top_features words as both rows index and
columns index
- Window of neighobr is 5
- populate Key_feature_list, top tfidf text
- for each and every value of rows of EssayTitle
    - put all the words of cell, as a list, andput it in CorpusList
    - calculate rangelength of CorpusList
    - Call cal_occ function with above values
-print the Co-occurance matrix
```

- Run TruncatedSVD with n_component = 500
- draw the plot between cum_var_explained and n_components
- choose best n_components (here 100)
- Use TruncateSVD, to reduce the dimensionality of matrix.
- take the top 2000 words, in glove_word, inorder to chech, against the word of Essay and project_title.
    - if the word exist, then we need to derive the avgW2V.
- merge via hstack, 'said' categorical , numerical features + project_title(avgW2V) + preprocessed_essay (avgW2V)
- Fit a model on on train (on above merge features) data by using GridSearchCV(XGBoostClassifier(eval_metric = 'auc', num_class = 2, nthread = 4)
- take the mean_train and mean-test value from the above fit model.
- Draw HEATMAP for both train and test data, between param_num_boost_round, param_max_depth and AUC(mean_train/test_score).
- Choose best max_depth and num_boost_round from best*param* function
- Draw roc_auc graph
- Create Confusion matrix, in heatmap